Universidade de Aveiro

2016

Departamento de Eletrónica, Telecomunicações e Informática (DETI).
Departamento de Engenharia Mecânica (DEM)

Amaro José Diaz da Costa

EQUIPAMENTO STANDARD PARA MONTAGEM DE COMPONENTES

Universidade de Aveiro

2016

Departamento de Eletrónica, Telecomunicações e Informática (DETI).
Departamento de Engenharia Mecânica (DEM)

Amaro José Diaz da Costa

EQUIPAMENTO STANDARD PARA MONTAGEM DE COMPONENTES

Relatório de projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Automação Industrial, realizada sob a orientação científica do Doutor Jorge Augusto Fernandes Ferreira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro.

O júri / The jury

Presidente / President	Prof. Doutor Pedro Nicolau Faria da Fonseca Professor auxiliar da Universidade de Aveiro
Vogais / Committee	Prof. Doutor Jorge Augusto Fernandes Ferreira Professor auxiliar da Universidade de Aveiro
	Prof. Doutor José Luis Costa Pinto de Azevedo Professor auxiliar da Universidade de Aveiro

Agradecimentos / Acknowledgements

O espaço limitado desta secção de agradecimentos, é com toda a certeza insuficiente, não sendo possível agradecer, como devia, a todas as pessoas que, ao longo do meu Mestrado em Engenharia de Automação Industrial me ajudaram, direta ou indiretamente, a cumprir os meus objetivos e a realizar mais esta etapa da minha formação académica.

Desta forma, é com um profundo sentimento de gratidão que deixo algumas palavras de apreço.

Em primeiro lugar agradeço a toda a equipa da empresa “Siroco S.A.”, pela disponibilidade, apoio na realização do projeto e por me terem recebido como um elemento integrante da mesma. Um especial agradecimento ao Eng.º Edmilson Cardoso por se mostrar um mentor no universo da automação.

Agradeço também ao meu orientador, Prof. Doutor Jorge Ferreira, pela sua disponibilidade, experiência, boa disposição, motivação, sugestões e críticas que foram sem dúvida uma mais valia na realização deste projeto.

Os ensinamentos técnicos, científicos e humanos que foram transmitidos pelo meu coorientador Prof. Doutor Vítor Santos ao longo de todo o mestrado e projeto desenvolvido, foram essenciais para a concretização desta etapa académica.

Quero também agradecer a todos aqueles que participaram no meu percurso académico e, que de alguma forma contribuíram e permitiram que fosse possível realizar e concluir o curso de mestrado.

Para começar agradeço à minha mãe (Mulher mais forte do mundo!), que pelas suas inúmeras qualidades humanas, sempre me incentivou, apoiou e orientou, para que este objetivo pessoal fosse atingido. Aos meus irmãos e primos, pelo seu estímulo, ânimo e carinho.

Os meus tios Manuel Carneiro e Josefa Diaz, foram sem dúvida pilares fundamentais no meu crescimento, ficando assim eternamente grato pelo apoio em todos os momentos da minha vida.

Às minhas tias Donna Diaz e Rosemarie Diaz e primos Habeeb Nuñez e Anthony Nuñez, por terem tido um papel bastante importante, na superação dos vários obstáculos.

Não podia deixar de mencionar, os meus amigos e companheiros, João Reis e Filipa Araújo, pelas suas sugestões e opiniões que contribuíram para este projeto.

Por fim, à minha namorada Dina Silva, quero também deixar o meu profundo carinho e agradecimento, por me ter dado força, por ter estado sempre ao meu lado, e pelo incentivo especialmente durante a realização deste projeto.

palavras-chave

Controladores Lógicos Programáveis, PLC, HMI, Programação Ladder, Grafcet, Blocos Funcionais, Programação Orientada por Objetos, Consola de operação.

resumo

O projeto apresentado surge da parceria entre a empresa Siroco e a Universidade de Aveiro, e descreve o estudo e a implementação de uma metodologia para programação de autómatos (*PLC*) e consolas de operação (HMI). Este documento descreve a proposta de uma metodologia baseada nos princípios da programação orientada por objetos que permite o desenvolvimento de blocos funcionais a usar na programação de alguns componentes típicos de um equipamento de montagem automatizado. Procura-se com esta abordagem diminuir o tempo de desenvolvimento de projeto dos sistemas tratados, e simplificar a implementação, integração, teste e manutenção das aplicações de *software*. Os blocos funcionais foram desenvolvidos na ferramenta de programação de PLC e HMI da Omron (*Sysmac Studio*), com recurso à utilização de estruturas denominadas de *Function* e *Function Block*.

A metodologia criada foi aplicada no desenvolvimento dos blocos funcionais propostos pela empresa Siroco, para os quais a empresa propôs que se elaborassem algoritmos de controlo para as estruturas mais importantes dos equipamentos desenvolvidos pela empresa, nomeadamente os cilindros pneumáticos, pratos rotativos e estações de trabalho.

É descrito o método utilizado para o desenvolvimento dos algoritmos, e é demonstrado como estes blocos podem ser aplicados no programa de controlo de um equipamento.

Para finalizar, são implementados dois casos de estudo que servem para demonstrar e validar a viabilidade da metodologia proposta neste projeto.

keywords

Logical Programmable Controllers, PLC, HMI, Ladder Programming, Grafcet, Function Blocks, Object-Oriented Programming, Operations Console

abstract

This project results from the partnership between the Siroco company and the University of Aveiro, and demonstrates the study and implementation of a methodology for the development of Programmable Logic Controller (PLC) and operator panels (HMI) programs. This document describes the proposed approach, based on principles of object-oriented programming that allows the development of functional blocks in the programming of some typical components of an automated assembly equipment. The propose of this approach is to reduce project development time of the treated systems and simplify the implementation, integration, testing and maintenance of software applications. The functional blocks were developed in the *Omron* PLC and HMI programming tool (*Sysmac Studio*), with the use of structures called of *Function* and *Function Blocks*.

The methodology created was applied in the development of functional blocks proposed by Siroco company, which are to design control algorithms for the most important structures of the typical equipment developed by the company, including pneumatic cylinders, indexing tables and workstations.

The method used to develop the algorithm is described, and it is shown how these blocks can be implemented in the control software of an equipment.

Finally, two case studies were implemented, which served to demonstrate and validate the feasibility of the methodology proposed in this project.

Índice

Índice de Figuras

Lista de Acrónimos

CAN – *Controller Area Network*

F – *Function*

FB – *Function Block*

FBD – *Function Block Diagram*

GRAFCET – *Graphe Fonctionnel de Commande, Etapes Transition*

HMI – *Human-Machine Interface*

HV – *High Voltage*

I/O – *Input/output*

LD – *Ladder Diagrams*

IEC – *International Electrotechnical Commission*

IL – *Instruction List*

PCB – *Printed Circuit Board*

PLC – *Programmable Logic Controller*

POU – *Program Organization Unit*

SFC – *Sequential Function Chart*

ST – *Structured Text*

1. INTRODUÇÃO

1.1. Enquadramento do projeto

A crescente competitividade e globalização do mercado obrigam a novos desafios e conceitos para os sistemas de produção, com vista a aumentar a modularidade, adaptabilidade, flexibilidade e capacidade de integração. O desenvolvimento de equipamentos para montagem de componentes é uma tarefa que requer conhecimentos relativos a várias áreas de desenvolvimento tecnológico, incluindo a engenharia mecânica, a engenharia eletrotécnica e a engenharia de automação industrial, uma vez que estes equipamentos são compostos por sistemas de atuação mecânica e diversos sensores para intervir e monitorizar os processos para o controlo e supervisão de movimentos e, por fim, por equipamentos de controlo.

No desenvolvimento de novos projetos de máquinas, procura-se usar partes de algoritmos que já tenham sido desenvolvidos e que sejam, de alguma forma, semelhantes ou que possam servir como base ao novo projeto. No entanto, é difícil tirar proveito de projetos anteriores visto que isso geralmente acontece porque as aplicações de desenvolvimento de algoritmos dos PLCs (*Programmable Logic Controller*) ainda não se encontram tão desenvolvidos, comparados com outras linguagens de alto nível. É um facto que os PLCs dependiam de linguagens pouco poderosas e além disso, a exportação e importação de código entre aplicações de desenvolvimento diferentes nem sempre é possível.

Com a demanda de novas linguagens e ferramentas para o desenvolvimento de algoritmos em ambiente de programação para PLC, levou a que a comissão do IEC publicasse uma norma que incluísse linguagens de programação de PLCs *standards*, esta norma é a IEC 61131.

Com a norma IEC 61131-3, passaram a existir estruturas programáveis, as quais vêm introduzir princípios da programação de computadores e que apresentam algumas caraterísticas interessantes em termos de *software* modular e reutilizável. Isto remete ao tema principal deste trabalho, que é como usar as potencialidades das novas linguagens e estruturas desta norma IEC 61131-3 para melhorar o desenvolvimento e manutenção de programas para PLC.

Este relatório descreve e discute a metodologia de desenvolvimento, com recurso a blocos funcionais de objetos reutilizáveis para PLC e HMI (*Human-Machine Interface*) no novo conceito de "Equipamento *Standard*". Estes objetos resultam da decomposição estrutural de uma máquina em partes mais pequenas (Estações de trabalho, robôs, etc.) em que, dependendo da sua complexidade, algumas podem ainda ser decompostas outra vez em mais objetos, cooperando assim de forma hierárquica em que cada uma delas estará associada a uma ação ou conjunto de ações particulares.

1.2. Empresa SIROCO

A Siroco é uma empresa sediada em Aveiro com atividade na área da automação industrial e robótica. É especialista no projeto e construção de equipamento industrial e ferramentas de precisão para a indústria que apresentam níveis elevados de automação com cumprimento integral da diretiva máquinas. A Siroco conta, entre outros, com clientes da indústria automóvel, aeroespacial, eletrónica, eletrodoméstico, madeira e o CERN.

1.3. Objetivo

Este projeto tem por objetivo realizar um estudo e implementar uma metodologia para a programação de autómatos (PLC) e consolas de operação (HMI), capaz de diminuir o tempo de desenvolvimento de projeto dos sistemas tratados, e simplificar a implementação, integração, teste e manutenção dos programas.

As tarefas propostas para este projeto foram a programação base da linha em PLC *Omron NJ* e de interface HMI para um novo conceito de equipamento *standard* para montagem de componentes. A finalidade destes programas é criar condições de adaptabilidade no que diz respeito à inserção e extração de componentes, respondendo em ambos os cenários de forma eficiente.

Em reunião com os elementos responsáveis da empresa, foram estabelecidos os seguintes objetivos:

- Desenvolvimento de um algoritmo de controlo de estações;
- Desenvolvimento de um algoritmo de controlo dos cilindros pneumáticos;
- Desenvolvimento de um algoritmo de controlo do prato rotativo;
- Integração dos alarmes/erros de utilizador utilizando o PLC NJ da *Omron*;
- Desenvolvimento de Interfaces HMI padrão para o controlo e monitorização das estações, cilindros, prato rotativo e de eventos.

Foi ainda definido que o ideal seria que o projeto fosse desenvolvido nas instalações da empresa, possibilitando uma maior e melhor interação com os responsáveis pelo projeto assim como com os demais colaboradores, conduzindo a uma maior aproximação entre o projeto planeado e o projeto a desenvolver.

1.4. Estrutura do relatório

O relatório está estruturado da seguinte forma: o capítulo 1 concerne à introdução ao trabalho e à apresentação dos objetivos do projeto.

No capítulo 2 é feita uma abordagem sucinta ao estado da arte, especificamente a composição genérica de um equipamento de montagem de componentes, comando e controlo e aspetos genéricos da modelação do controlador e HMI.

O capítulo 3 aborda a metodologia proposta para o desenvolvimento do projeto como também a implementação da metodologia apresentada.

No capítulo 4 serão demonstrados os casos de estudo que validam os objetivos do projeto e é feita a análise dos resultados obtidos.

O capítulo 5 completa uma apreciação ao trabalho desenvolvido neste projeto e são apresentadas algumas conclusões e propostas de trabalhos futuros.

2. ESTADO DA ARTE

O projeto a realizar enquadra-se, de uma forma genérica no sector secundário e, mais especificamente, no ramo da automação e robótica. Desta forma, procedeu-se a uma busca de literatura relativa ao tema, assim como algum tipo de trabalhos já desenvolvidos no contexto. Porém, constatou-se que a existência de trabalhos deste tema era diminuta ou pouco relacionada, facto este talvez explicável devido a extrema especificidade do tema, que por norma não é comum ser abordado em trabalhos académicos. Para além disso, esta temática é muitas vezes restrita ao ambiente interno de cada empresa.

No entanto, o conceito de modularização dos componentes de automação, sejam eles de *hardware* e/ou de *software*, é apresentada em alguns artigos científicos [1] [2] como sendo uma técnica que apresenta um elevado potencial para, por exemplo, minimizar custos internos de desenvolvimento/criação de máquinas automatizadas. A nível externo, este conceito poderá ser uma grande oportunidade a nível de mercado, pois tendo em conta que os seus custos internos de conceção e produção seriam menores, ao entrar no mercado, seria também com um preço mais competitivo em relação aos restantes.

No Estado da Arte do presente trabalho, será feita uma abordagem da composição genérica de um equipamento de montagem de componentes, onde as tarefas a automatizar em equipamentos deste tipo envolvem a utilização de diversos tipos de equipamentos. A abordagem a este processo será feita através das seguintes subsecções:

- **Equipamento de produção:** transportadores automatizados, linhas de montagem e estações de trabalho de vários tipos e funções;
- **Controladores e periféricos:** computadores pessoais, controladores lógicos programáveis (PLC), controladores embebidos noutros equipamentos, periféricos de interface com o utilizador, sensores e atuadores, etc.;
- **Redes locais:** usando diversos protocolos e suportes físicos para ligar os vários computadores e sistemas computadorizados;

Por fim será feito uma abordagem sobre metodologias de programação de PLC e HMI.

2.1. Composição estrutural genérica de equipamentos de montagem de componentes

Como parte fulcral do trabalho é necessário fazer um estudo e uma descrição genérica dos sistemas automatizados mais relevantes que integram o panorama geral de uma linha de montagem típica. Sendo este o sistema que irá ser tratado na área de investigação e desenvolvimento neste projeto, apresenta-se, nos parágrafos que se seguem, o seguinte sistema estudado.

Um sistema de montagem, ou linha de montagem, é um meio utilizado para a produção em massa e em grande escala que possui um nível de automatização elevada, com o objetivo de aumentar a eficiência em sistemas de produção. São tipicamente construídos em torno de um mecanismo de

transporte para mover as peças, que normalmente são colocadas numa palete e são transportadas ao longo do meio de transporte, por entre as várias estações de trabalho onde as operações de montagem ocorrem. Estas operações podem incluir a assemblagem de peças, onde poder-se-á efetuar testes elétricos ou verificações visuais usando visão artificial para garantir que a montagem final se encontra de acordo com os padrões exigidos, entre outras operações.

Estes sistemas são constituídos por meios de transporte que podem ser lineares ou circulares e, por norma, possuem um ponto de abastecimento num dos extremos da linha e o ponto para retirar o produto no outro extremo da linha. Em alternativa, existem linhas continuas em que pode ser usada a técnica de "*loop*" em que o produto poderá percorrer a linha de montagem mais do que uma vez [3].

Nesta última abordagem é muito simples incluir um conjunto de estações de trabalho com tarefas específicas onde se executam tarefas manualmente ou por meio de um equipamento automático, em decorrência da complexidade ou custo da automação. As estações de trabalho podem executar processos de modificação de forma e superfície, colagem, soldagem, entre outros. Resumindo, cada estação executa uma determinada operação de forma que, no final, o conjunto das operações de todas as estações são as necessárias para completar uma peça acabada. Numa configuração mais simples, o número de peças no sistema em qualquer momento é igual ao número de estações. Porém, nas configurações mais complexas, o número de peças pode ser superior ao número de estações.

A Figura 1 ilustra uma configuração típica de uma linha de processamento automatizada onde múltiplas estações de trabalho estão interligadas por um tapete rolante também automatizado.

A Figura 2 apresenta uma configuração típica de um sistema de montagem automatizada, onde é executada uma sequência de operações de forma a combinar múltiplos componentes numa única entidade. Esta entidade poderá ser um produto final ou uma parte de uma montagem de um produto maior [4].

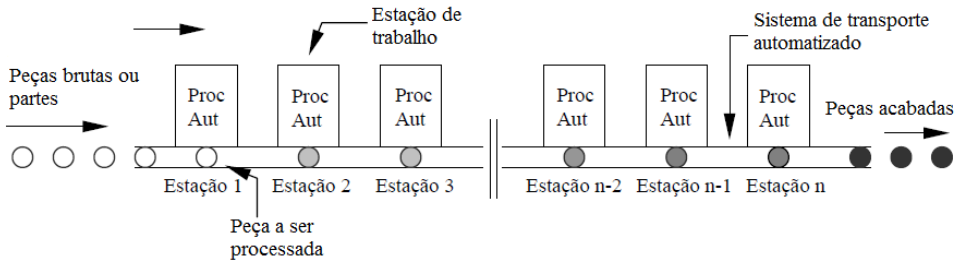


Figura 1 - Configuração geral de linhas de produção automatizadas [4].

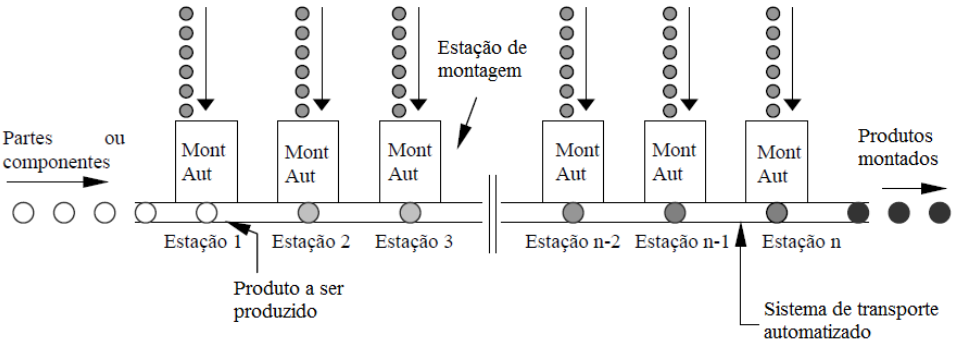


Figura 2 - Configuração geral de sistemas de montagem automatizados [4].

A operação da linha de produção pode ser síncrona ou assíncrona. No primeiro caso, todas as estações ficam à espera que a estação mais lenta termine o processo antes que as peças possam ser transferidas para a estação seguinte. No segundo caso, os processos das estações são independentes umas das outras, isto é, o sistema de transporte está sempre ativo e, sempre que um processo é terminado numa estação, a peça é libertada para o meio de transporte e a peça segue para a estação seguinte caso esta esteja livre, caso contrário fica em espera. Desta forma, permite que esta estação anterior fique disponível para receber outra peça, otimizando assim o processo.

Existem vários tipos de linhas de produção, capazes de produzir vários lotes do mesmo modelo ou grupos de modelos similares. Os diferentes tipos de linhas de montagem estão ilustrados na Figura 3, onde diferentes modelos estão representados por diferentes formas geométricas [5] [6].

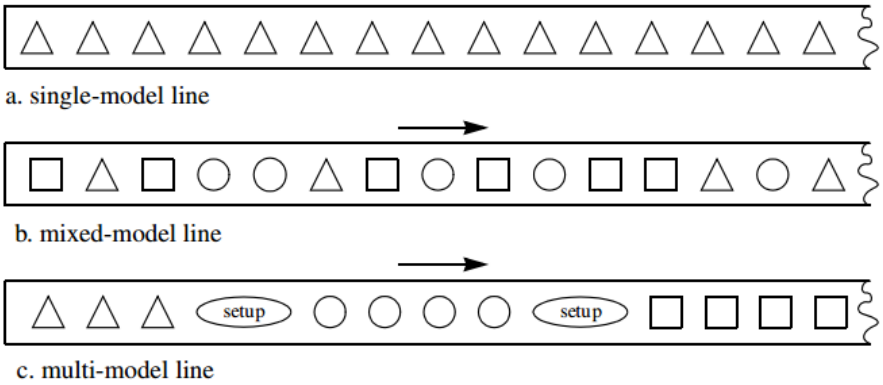


Figura 3- Tipos de linha de produção [7].

Muitos sistemas de montagem mais dedicados são construídos em torno de um prato rotativo em vez de um sistema de transporte linear. Estas linhas de transporte circulares definem uma superfície plana, normalmente com a configuração duma mesa circular e, em redor da mesa, existem ninhos onde se alojam as peças. As estações de trabalho são estacionárias e usualmente localizadas na periferia da mesa (ver Figura 4). O prato roda de forma indexada, de forma a orientar os ninhos na posição correta para o processamento das estações de trabalho. Quando as estações terminam o processo, o prato volta a rodar, orientando as peças com a próxima estação de trabalho.

O controlo de um prato rotativo pode ser feito diretamente por um meio de controlo que esteja a ser implementado pelo responsável do desenvolvimento do sistema de montagem ou por uma carta de controlo específica do fornecedor. Estas cartas podem comunicar com outros meios de controlo. Este último método acaba por se tornar mais utilizado, uma vez que a parte de controlo já se encontra desenvolvida, tornando assim a aplicação de um prato rotativo numa linha de montagem uma tarefa mais simples [4].

Neste último método, se quisermos efetuar um movimento, basta enviar um comando de rotação ao controlador do prato rotativo e este inicia a rotação para a posição seguinte. Quando o prato chegar à posição, o controlador retorna um sinal que indica que o prato rotativo já efetuou a rotação e que se encontra na posição correta, caso contrário, envia um sinal de erro.

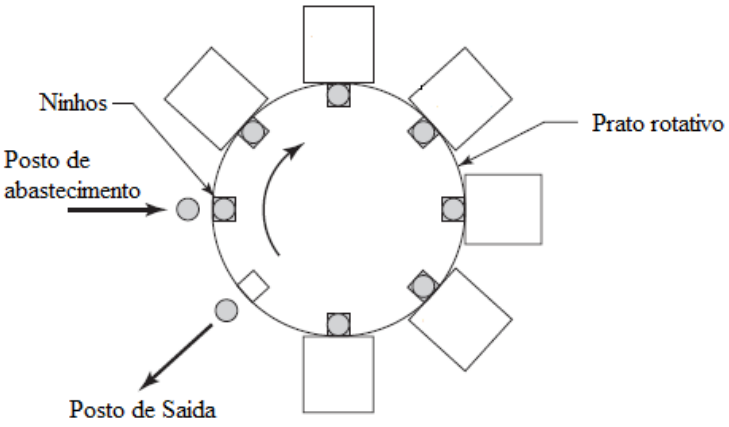


Figura 4 - Representação genérica de um prato rotativo [4].

2.2. Principais elementos de um sistema de controlo e comando

Nesta secção abordam-se, com algum detalhe, alguns destes equipamentos, nomeadamente: controladores, sensores e atuadores [8].

2.2.1. Soluções de comando e controlo

O controlo lógico tem por objetivo complementar sistemas lógicos de maneira que eles respondam a eventos externos ou internos de acordo e conforme o desejado de um ponto de vista utilitário. O controlo lógico é um meio de automatização que se realiza por meio de circuitos (elétricos, hidráulicos, pneumáticos, etc.) em que as variáveis são binárias (valor 0 ou 1) [8].

Os sistemas de controlo e comando de um sistema de automação proporcionam uma série de funções-chaves, tais como:

- Controlo global dos elementos dum sistema para assegurar que todos operam como planeado e na sequência prevista;
- Apresentação de dados relativos a um sistema ou processo, para o utilizador, supervisor e/ou para um processo supervisor.
- Despoletar alarmes, erros e informação para identificar falhas do sistema.

- Funcionalidades de segurança

Computadores para controlo de processos

O uso de computadores digitais para o controlo industrial de processos, surgiu nos processos contínuos industriais na década de 1950. Antes disso, os controladores analógicos eram usados para implementar o controlo contínuo e os sistemas controlados por relés eram usados para implementar o controlo discreto. Nesse período, a tecnologia dos computadores ainda estava muito pouco evoluída e os computadores existentes para o controlo de processos industriais, eram muito volumosos e caros; comparados com a tecnologia atual, os computadores da década de 1950 eram lentos e pouco fiáveis. Para além de que os computadores instalados, muitas vezes, custavam mais do que o processo que iam controlar. Os avanços tecnológicos na área dos circuitos integrados levaram ao desenvolvimento de microprocessadores. Atualmente, os processos industriais são controlados por computadores digitais baseados em microprocessadores.

Existem várias maneiras para controlar processos com um computador. A Figura 5 mostra a distinção entre a monitorização de processos e o controlo de processos. Na monitorização de processos, o computador é usado simplesmente para recolher dados do processo (a), enquanto no controlo do processo, de acordo com a nomenclatura, o computador controla o processo (b). Em alguns casos, o processo pode ser controlado em malha aberta, em outros o processo precisa de algum tipo de retorno para assegurar que as instruções de controlo foram corretamente realizadas. Esta situação mais comum é o controlo em malha fechada (c).

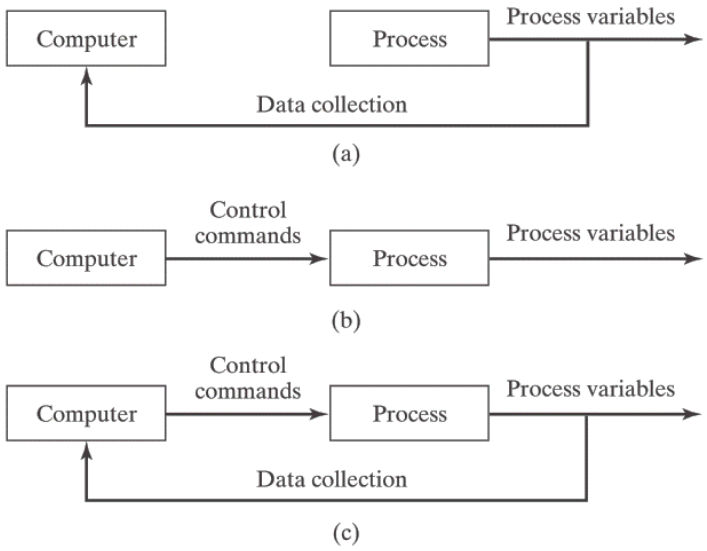


Figura 5 - Formas de controlar processos através de um computador [4].

Autómatos programáveis

Os Autómatos programáveis, vulgarmente designados PLC, são elementos fundamentais dos modernos sistemas de automação industrial. O PLC é um equipamento eletrónico, que permite a automatização de sistemas, sendo, um dos controladores mais utilizados na indústria, graças à sua multifuncionalidade e flexibilidade de programação. Um PLC automatiza uma grande quantidade de ações (de acordo com as necessidades) com precisão, confiabilidade, rapidez e pouco investimento. São analisadas informações de entradas, processa-se, decisões são tomadas; tudo isto é efetuado em

simultâneo com o desenrolar do processo. Cada PLC possui múltiplas unidades de processamento permite que vários programas estejam a ser executados em simultâneo no controlador. Os programas diferem nos níveis de prioridade e tipo de execução (periódico/cíclico ou por interrupções).

Mas nem sempre foi assim, visto que no início os PLCs pretendiam ser uma alternativa mais flexível à lógica elétrica e baseada em *timers*, que era comum encontrar nos painéis de controlo. Com a sua popularidade e facilidade de programação, os PLCs evoluíram muito desde o seu aparecimento (década de 1970). Hoje existem vários tipos de PLC, sendo os mais poderosos computadores industriais de elevada desempenho. De qualquer forma, todos os PLC têm capacidades elementares como: Temporizadores, Contadores, Registos, Operações lógicas elementares, várias entradas e saídas digitais e analógicas, expansíveis com módulos de IO [8] [9].

Os PLCs são atualmente equipamentos imprescindíveis nos processos indústrias, possuindo algumas características como [8] [10]:

- Microprocessadores avançados e tecnologia eletrónica permitem de o PLC efetue ciclos de leitura mais rápidos;
- Simplificação nos quadros e painéis elétricos. Toda a cablagem do comando fica resumida a um conjunto de entradas e saídas. Como consequência, qualquer alteração necessária torna-se mais rápida e barata.
- Robustez adequada aos ambientes industriais;
- Programação por meio de computadores pessoais (PC);
- Linguagens de programação de alto nível.
- Funções avançadas. Os controladores podem realizar uma grande variedade de tarefas de controlo através de funções matemáticas, controlo de qualidade e informações para relatórios.

À semelhança dos computadores, um PLC segue o modelo de um sistema que engloba "*hardware*", "*software*", dados e procedimentos. De forma geral, os elementos básicos que constituem o modelo de um PLC são os seguintes [11]:

- Unidade Central de Processamento (CPU);
- Sistema Operativo (SO);
- Memória de programa e dados;
- Entradas (*Inputs*) e saídas (*Outputs*);
- Comunicações;
- Alimentação.

A Figura 6 ilustra o esquema básico da estrutura de hardware de um PLC.

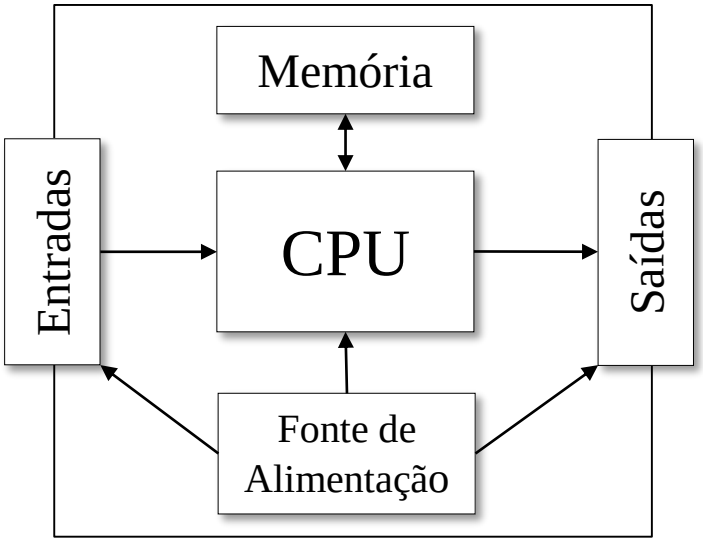


Figura 6 - Esquema da estrutura de *hardware* de um PLC – Baseado em [12].

Quando se inicializa um PLC, este cumpre uma rotina de operações pré-programadas. O controlador começa por executar uma série de verificações, tal como o funcionamento do CPU, das memórias, o reconhecimento dos módulos de entrada e de saída ligadas ao PLC, desativa todas as saídas, verifica a existência de algum programa de utilizador e, no final, emite um aviso de erro caso algum dos itens acima falhe [13].

No início de cada ciclo de execução são lidas as entradas que possam provir de um qualquer processo e, de seguida o resultado desta verificação é armazenado na memória de dados de processamento. Os sinais de entrada podem ser digitais ou analógicos e podendo, ainda, ser usados vários módulos de entradas que se adequem às necessidades do sistema. As entradas são meios de comunicação do PLC com o processo a ser controlado. As entradas recebem os sinais dos sensores e transforma-os em sinais digitais para serem processados pelo CPU [13].

A memória do programa e a memória utilizada para a transferência de dados estão separadas pois este facto resulta na necessidade de armazenar os programas do PLC, mesmo com a alimentação desligada. Na execução do programa, o CPU acede à memória do autómato para armazenamento e transferência de dados [13].

No final de cada ciclo as saídas são atuadas de acordo com as entradas e o programa de controlo, de modo a comandar o equipamento ou máquinas envolvidas no processo [13].

Inicia-se então um novo ciclo do PLC, repetindo todo o processo, de acordo com a Figura 7.

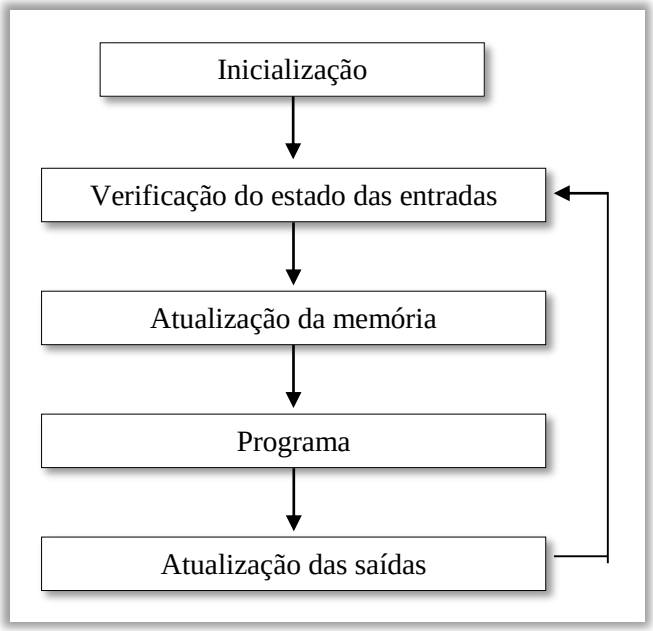


Figura 7 - Ciclo de processamento de um PLC.

Os PLCs possuem portas de comunicação que podem ser do tipo: portas série (RS232 e RS485), redes de campo (CAN, *Profibus*, *DeviceNet*, etc.), portas *Ethernet* (TCP/IP), entre outros. Os autómatos programáveis são sistemas de controlo que atualmente são capazes de comunicar com outros sistemas de controlo, fornecer relatórios de produção, calendarização da produção, e diagnosticar as suas próprias falhas e os da máquina ou processos que controlam [14] [15].

A programação de controladores compatíveis com a norma IEC 61131-3 é feita através das cinco linguagens definidas pela mesma. Atualmente diversos PLCs disponíveis no mercado têm boa compatibilidade com a norma. As cinco linguagens que definem esta norma são [16]:

- SFC: Sequential Function Chart
- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram

A norma 61131-3 em sua parte tem por objetivo que na programação dos controladores a abordagem e estruturação seja fundamentada em três princípios: modularização, estruturação e reutilização de código [16].

Os ambientes de programação compatíveis com a norma IEC permitem a utilização de linguagens de programação diferentes dentro do mesmo programa, de forma que cada problema de controlo seja tratado pela linguagem que mais se adequa, dentro dessas considerações. A norma IEC não restringe nenhuma linguagem em relação a elementos comuns, que é a parte mais importante da norma. Ou seja, o utilizador pode usar a linguagem da sua preferência para utilizar todos os elementos comuns [16].

Refira-se que existem no mercado várias opções de marcas com caraterísticas semelhantes (Omron, Siemens, Beckhoff, Schneider, GE, Allen-Bradley, etc.) entre as quais aplicam-se os mesmos conceitos apresentados neste capítulo.

Sistemas embebidos

Os sistemas embebidos são usualmente encontrados em aplicações do dia-a-dia, contudo, também podem ser encontradas em sistemas de automação complexos e sistemas de controlo adaptáveis. Este tipo de sistemas quando comparados com outros tipos de controladores industriais, apresentam normalmente consumos elétricos mais baixos, mas em contrapartida são mais limitados em espaço de memória. No entanto, para resolver sistemas em tempo real, os sistemas embebidos são mais baratos e muito mais fáceis de conceber. A simplicidade no desenvolvimento de soluções, reflete-se a termos de *hardware* e *software* [17].

Os sistemas embebidos são normalmente programados para realizar funcionalidades em tempo real. O *hardware* de processamento normalmente utilizado inclui microcontroladores, microprocessadores, FPGAs (*Field Programmable Gate Arrays*), DSPs (*Digital Signal Processors*) e ASICs (*Application-Specific Integrated Circuits*), cada um com as suas caraterísticas específicas [17].

PC Industrial

O PC industrial tem surgido com mais frequência em sistemas industriais. O interesse em aplicar este tipo de computadores em processos de controlo é maior, isto deve-se ao facto destes equipamentos serem cada vez mais fiáveis, e apresentarem garantias de resultados consistentes. Computadores modernos podem executar de forma confiável tarefas como: uniformidade no controlo, proporcionar grandes velocidades nos processos de controlo e executar vários processos em simultâneo [18].

Interface homem-máquina (HMI)

Na automação industrial, uma HMI é uma interface gráfica que permite ao utilizador interagir com uma máquina. O objetivo do diálogo Homem-máquina é permitir a comunicação entre o Homem e a máquina de um modo simples. Assim, a HMI está normalmente próxima da linha de produção, instalada na estação de trabalho, traduzindo os sinais vindos do PLC para sinais gráficos de fácil entendimento.

Uma HMI é um *hardware* industrial composto normalmente por uma tela tátil e um conjunto de teclas para navegação ou inserção de dados e utiliza *software* proprietário para sua programação.

Existem várias aplicações onde se podem utilizar uma HMI, como por exemplo:

- Visualização de alarmes gerados por alguma condição anormal do sistema;
- Visualização de dados dos equipamentos de uma linha de produção;
- Visualização de dados de processo da máquina;

- Alteração de parâmetros do processo;
- Operação em modo manual de componentes da máquina;
- Alteração de configurações de equipamentos.

Deste modo, o utilizador tem uma maior facilidade de análise de dados e diagnóstico físico da máquina, podendo assim evitar erros de atuação [10] [19].

***Supervisory control and data acquisition* (SCADA)**

O processo de monitorização é muito importante, existindo, por isso, no mercado vários produtos capazes de realizar de forma simples essa função e que funcionam com a generalidade dos controladores industriais. Um sistema de supervisionamento típico é capaz de adquirir e transmitir dados para um processo uma interface Homem-máquina (HMI) que permite ao operador enviar comandos e ler os dados adquiridos do processo, unidades terminais remotas (RTU – *Remote Terminal Units*) que estão ligados diretamente aos processos para controlo e aquisição de dados, e por fim, uma rede de comunicação que liga o computador central aos RTUs [4] [8]. A relação entre o controlo de supervisão e o controlo do processo é ilustrado na Figura 8.

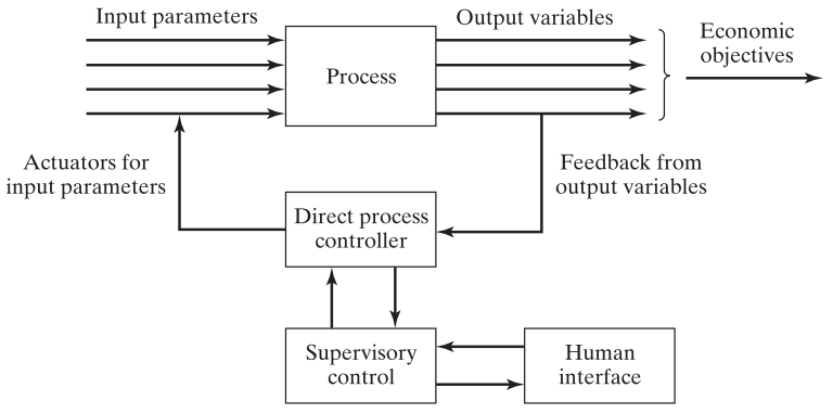


Figura 8 - Controlo de supervisão sobreposto a outros sistemas de controlo [4].

2.2.2. Sensores e detetores industriais

Os sensores são componentes indispensáveis em qualquer sistema automatizado. São eles os responsáveis por obter informação do funcionamento dos sistemas e da evolução dos processos a ser controlados. Estão, por isso, associados a funções de supervisão, monitorização. Em termos industriais, existe uma grande variedade de sensores, adaptados a determinadas funções e grandezas a medir. Neste subcapítulo serão descritos alguns dos sensores mais aplicados na indústria [14].

Detetores de proximidade: de contacto, óticos, indutivos, capacitivos, magnéticos, entre outros.

Sensores analógicos: de temperatura, carga, resistência, humidade, imagem, entre outros.

2.2.3. Atuadores industriais

Os atuadores são fundamentais em qualquer sistema de controlo industrial. Representam a saída do sistema, isto é, a possibilidade de atuar nos processos ou equipamentos a que está associado. Existe uma enorme variedade de tecnologias associadas a estes componentes, abordando em particular os atuadores pneumáticos, hidráulicos e os eletromagnéticos dada a sua importância na indústria.

Os atuadores podem ser do tipo hidráulico, pneumático e eletromecânicos. A tecnologia do ar comprimido apresenta uma maneira fácil e segura de obter o movimento linear. A utilização de mecanismos intermédios permite transformar esse movimento linear em movimento de rotação. Dentro deste tipo de atuadores, existem os cilindros pneumáticos e motores pneumáticos. Os atuadores hidráulicos comportam-se de forma semelhante aos pneumáticos, mas neste caso o fluido utilizado é óleo.

Os atuadores eletromecânicos são essencialmente implementados pelos motores elétricos. Estes equipamentos são muito usados na indústria devido ao facto de serem mais baratos, quando comparados com os motores pneumáticos e hidráulicos. São fáceis de comandar, adaptam-se razoavelmente bem às cargas a que são sujeitas e apresentam rendimentos elevados. Os motores elétricos mais representativos são de dois tipos: Motor de corrente contínua (CC), Motor de corrente alternada (CA) e Motor de passo a passo. Os motores de corrente contínua eram os mais populares até há pouco tempo, mas com o, aparecimento e generalização de variadores de velocidade eletrónicos, vieram facilitar controlo de motores de corrente alternada. Atualmente passou a ser mais usual aplicar motores de corrente alternada, por serem mais baratos e possuírem custos de instalação e manutenção mais baixos do que os motores de corrente contínua [14].

2.3. Estratégias de desenvolvimento e metodologias de programação de PLC

Nesta secção será feito uma referência ao *software* como sendo uma ferramenta industrial e serão também discutidos os métodos, técnicas e ferramentas de desenvolvimento que podem melhorar a produtividade e qualidade do *software* desenvolvido.

2.3.1. Estratégias de desenvolvimento de *Software*

Hoje em dia é comum haver uma aceitação de que programadores do presente e passado se envolvam para perceber a importância da disciplina no desenvolvimento de código, modularidade e reutilização de componentes de *software*.

Ainda que os programadores estejam interessados em ferramentas de simulação e teste, ainda existe algum ceticismo em relação à adoção de técnicas de abstração. É comum aos profissionais ligados a esta área de desenvolvimento estarem diariamente envolvidos em situações de esforço para entregar e ajustar código em tempo útil e como consequência disso, na maioria das vezes, os programadores estão mais interessados em seguir meios que permitam concretizar em curto prazo do que seguirem métodos que garantam resultados não só a curto ou médio prazo mas também a longo prazo. O uso de métodos impróprios é visto como um desperdício de tempo e resultam em problemas evidentes nos algoritmos desenvolvidos, como por exemplo a carência de uma estrutura sólida,

dificuldade de integração de algoritmos que tenham sido desenvolvidos por mais do que um programador, dificuldade de reutilização e manutenção do mesmo código.

Os trabalhos desenvolvidos pelos programadores, não costumam ser sujeitos ao controlo de qualidade. As medidas comumente aceites são linhas de código, número de erros, *debugging*, etc. Este facto resulta tipicamente na ideia de que o importante é que o *software* funcione sem qualquer consideração na qualidade intrínseca do código. O facto de que o *software* funciona não garante que possa ser facilmente modificado ou atualizado para enfrentar novas exigências e, definitivamente, não assegura a sua reutilização. Por isso, estas práticas não são desejáveis e resultam num sério obstáculo no que se entende por ser a boa prática de desenvolvimento de *software*. Se não se conseguir mudar este tipo de abordagem e o sistema de valores subjacentes, não existe possibilidade de introduzir metodologias de desenvolvimento corretas e sistemas de garantia de qualidade [16].

Fatores de qualidade de *software*

Os fatores de qualidade de um *software* podem ser divididos em dois grupos: as qualidades externas, que são aquelas reconhecidas pelo utilizador, e as qualidades internas, que estão relacionadas com as caraterísticas intrínsecas do *software* que são fundamentais para a obtenção das qualidades externas [16].

As qualidades externas principais são a exatidão, a performance e a facilidade de usar o *software*. Estes são os três requisitos mais gerais que o utilizador final é capaz de perceber e avaliar.

O termo exatidão pode ter vários significados. O primeiro refere-se ao número de erros presentes no *software*. Um programa é mais exato quanto mais estiver livre de erros ou, melhor dizendo, se durante a sua utilização muito poucas falhas ocorrem. Uma interpretação ligeiramente diferente tem a ver com a confiabilidade, medindo a correspondência entre o esperado e o comportamento atual do *software*. Outro significado que pode estar relacionado com exatidão é robustez. Um *software* é robusto se dada a ocorrência de funcionamento imprevisíveis, o comportamento resultante não falhe, ou que pelo menos não traga consequências negativas para o sistema controlado.

À parte da exatidão, a performance é geralmente muito importante. De modo geral, podemos dizer que um pacote de *software* de controlo oferece boas performances se o seu tempo de resposta é compatível com as necessidades do utilizador [16].

Outro fator externo que afeta a perceção da qualidade do *software* é a facilidade de utilização. A facilidade de utilização significa essencialmente que é necessário um período de aprendizagem curto para que o operador consiga operar no sistema sem dificuldades. Se considerarmos o caso dum *software* de PLC, o utilizador não é normalmente especialista em tecnologias de informação. Tipicamente tem um bom conhecimento do processo controlado e o seu papel é limitado a definir os parâmetros do sistema, operação inicial e procedimentos de operação. Como consequência, a facilidade de utilização é conseguida através de interfaces Homem-máquina claras e a disponibilidade de botões dedicados para reações rápidas.

Em aplicações tradicionais, a facilidade de utilização está relacionada com a disponibilidade de interfaces poderosas e avançadas. É certo que uma interface gráfica, que apresente muitos dados em simultâneo e capaz de oferecer muitas opções possíveis, aumenta a produtividade de um

utilizador experiente capaz de usar todas essas opções. Mas se o utilizador não estiver familiarizado com a interface é melhor usar soluções mais simples e recorrer a uma abordagem passo a passo [16].

As qualidades externas resultam da aplicação de técnicas e métodos adequados no projeto e desenvolvimento. Por outras palavras, as qualidades externas são asseguradas pelas qualidades internas. De seguida apresenta-se algumas das qualidades internas mais importantes [16]:

Compreensibilidade: uma caraterística chave para o *software*, que favorece correções, modificações e atualizações e possível reutilização. De um modo geral, a compreensibilidade do *software* é garantida ao se decompor por partes de tal maneira que cada parte pode ser descrita de forma independente das outras. A maioria das metodologias de projeto são concebidas para obter esta modularidade, porém, nem todas atingem o nível desejado de compreensibilidade [16].

Reutilização: os métodos de obter compreensibilidade por si só não implicam a capacidade de reutilização. Um pacote de *software* é reutilizável se este for independente do ambiente nativo e permanece independente do ambiente onde é sucessivamente integrado [16].

Verificabilidade: especialmente para sistemas em tempo real, a verificabilidade é outra qualidade interna importante. De um modo geral, um *software* é verificável se for fácil de testar de forma disciplinar. Além disso, um *software* pode ser verificado e testado num tempo razoavelmente curto se for possível verificar e testar separadamente cada uma das partes e depois verificar o seu conjunto [16].

Manutenção: verificabilidade é estritamente relacionada com manutenção; a possibilidade de aceder de forma fácil e isolada à parte que precisa ser modificada ou melhorada [16].

Isolamento: a independência mútua das partes individuais do *software* pode ser medida com respeito ao impacto de erros e para a execução de operações de manutenção. Relativamente aos aspetos anteriores, pode-se afirmar que o pacote apresenta um alto grau de isolamento se um problema que ocorra dentro de um módulo específico não tenha efeitos (ou poucos efeitos) sobre os outros módulos [16].

Sumariando, para que o *software* se torne uma verdadeira ferramenta industrial, são necessárias ferramentas e metodologias adequadas. Tais ferramentas e métodos devem garantir as qualidades internas de modo a criar condições para garantir as qualidades externas. Este parece ser o único meio eficaz para enfrentar a complexidade intrínseca do *software*. Como visto, a modularidade é o fator chave para a qualidade interna [16].

2.3.2. Metodologias e técnicas para a programação de PLCs

Num momento inicial, dá-se a conhecer as técnicas e metodologias tipicamente utilizadas para a implementação de software. Deste modo, é dado a conhecer a utilização de ferramentas de modelação como as máquinas de estados finitos, os *Petri Nets* e os GRAFCET.

No final, é feita uma breve descrição à programação orientada por objetos e às suas características interessantes do ponto de vista de desenvolvimento de *software* para PLC. São ainda apresentadas as plataformas que a norma IEC 61131-3 disponibiliza como base para a implementação dos programas de PLC.

Diagramas de Processo

Na implementação de um sistema de automação, o primeiro objetivo do projetista deve ser a transmissão eficiente e segura do conteúdo técnico dos processos industriais a serem automatizados. Há várias formas de representação gráfica de sistemas dinâmicos, onde a evolução depende de eventos que ocorrem no processo que são utilizados com frequência. Estes fluxogramas e diagramas são implementados por meio de símbolos padronizados que mostram as relações funcionais entre os seus diversos componentes, e, por meio desse meio gráfico, é possível entender as condições operacionais de um sistema [10].

Para apresentar os processos industriais, é comum usarem-se [10]:

- Diagramas de blocos;
- Diagramas de fluxo de processo;
- Diagramas de interligação e instrumentação (Diagrama de estados, *Petri Nets e* GRAFCET) ou fluxogramas de engenharia, P&ID (*Piping and Instrumention Diagrams*).

Todos os três são meios de comunicação complementares, indispensáveis para a comunicação entre diversos especialistas que colaboram nos processos e com os engenheiros de automação. Estas técnicas não pretendem minimizar as funções lógicas, pelo contrário, o seu fim reside na imposição de um funcionamento rigoroso, de forma a evitar incoerências, bloqueios, ou conflitos no funcionamento do programa. Nesse sentido, pretende-se assegurar, durante a fase de projeto e de desenvolvimento a elevada probabilidade de o programa estar isento de erros de conceção [10]. Existem diversas técnicas que permitem a modelação de sistemas de eventos discretos, tais como as, máquinas de estados finitos, *Petri Nets* e o GRAFCET.

Máquinas de estados finitos

Uma máquina de estados finitos é um sistema de transições que comporta um número finito de estados e de transições entre esses estados. Este método permite descrever de forma gráfica o sistema, no que diz respeito à sua evolução ao longo do tempo [12].

De entre as representações mais vulgarmente usadas, duas são referências especiais: os diagramas de estados e os fluxogramas. Ambas representações são usadas para a descrição de algoritmos [12].

Os diagramas de estados que são usados em muitas áreas de aplicação mostram os estados sob a forma de círculos e transições na forma de setas. Os arcos são etiquetados pela combinação de eventos e ações cuja verificação dessas condições permite alterar o estado do sistema [21]. (ver Figura 9).

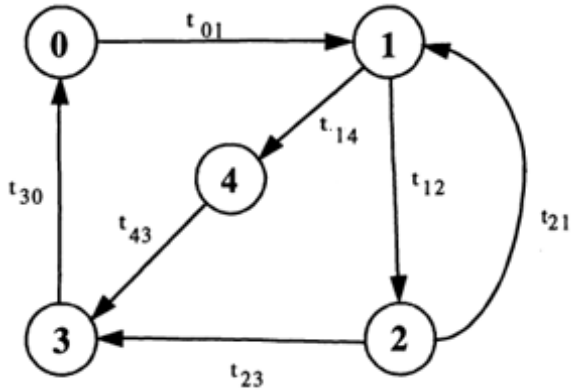


Figura 9 - Elementos de um diagrama de estados [21].

Os fluxogramas são mais vocacionados para a descrição de algoritmos e são, possivelmente, a ferramenta de modelização de *software* mais antiga e pode ser considerada de compreensão intuitiva. Este tipo de representação apresenta os estados em forma de retângulos e as transições entre os estados são representadas em forma de losangos, são detalhadas em termos das dependências de eventos de entrada. Contudo, existe o inconveniente de resultar num reduzido suporte às técnicas estruturadas de programação. O estado inicial e final (caso exista) é explicitamente modelizado [12]. A Figura 10 apresenta um exemplo de um fluxograma.

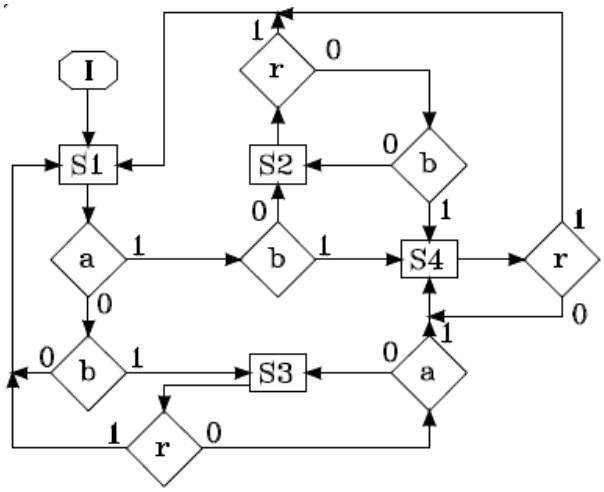


Figura 10 - Elementos de um fluxograma [18].

De um modo geral, as máquinas de estados finitos apresentam limitações quanto à modelização de sistemas complexos, nomeadamente a incapacidade de modelização de processos concorrentes, bem como a representações hierárquicas. As dificuldades de modelização de sistemas paralelos e de suportar uma decomposição hierárquica excluem a utilização da máquina de estados finitos [12]. No que toca à representação de sistemas mais complexos, a subsecção seguinte constitui uma resposta mais adequada para a redução destas limitações.

Redes de Petri (*Petri Nets*)

As Redes de Petri, embora possam ser encaradas como um formalismo de representação abstrato, revelam-se adequadas à representação das dependências referidas no parágrafo anterior. As redes de Petri foram propostas por Carl Adam Petri, que criou este método de estudo dos sistemas dinâmicos a eventos discretos na sua tese de doutoramento (1962). É uma ferramenta de modelização e análise de sistemas que, permite a construção do seu modelo de funcionamento. Esta metodologia permite modelizar sistemas complexos que apresentem caraterísticas de funcionamento concorrente e com necessidade de interatuar. [12].

As Redes de Petri (RdP) destacam-se na engenharia atual pelas seguintes qualidades [10]:

- Capturam as relações de precedência e os vínculos estruturais dos sistemas reais;
- São graficamente expressivas;
- Modelam conflitos e filas;
- Têm fundamento matemático e prático;
- Admitem várias especializações (RPs temporizadas, coloridas, estocásticas, de confiabilidade, etc.).

Uma Rede de Petri é um grafo orientado quem tem dois tipos de nós: transições e posições. Os arcos do grafo partem de algumas posições ou vice-versa. Aos arcos associam-se números fixos. Cada posição pode conter um número de marcas e estas, sob certas condições, podem mover-se ao longo dos arcos, respeitados os sentidos destes [10] (ver Figura 11), Este método permite descrever de forma gráfica tantos processos sequenciais e especialmente processos não sequenciais (assíncronos) [21].

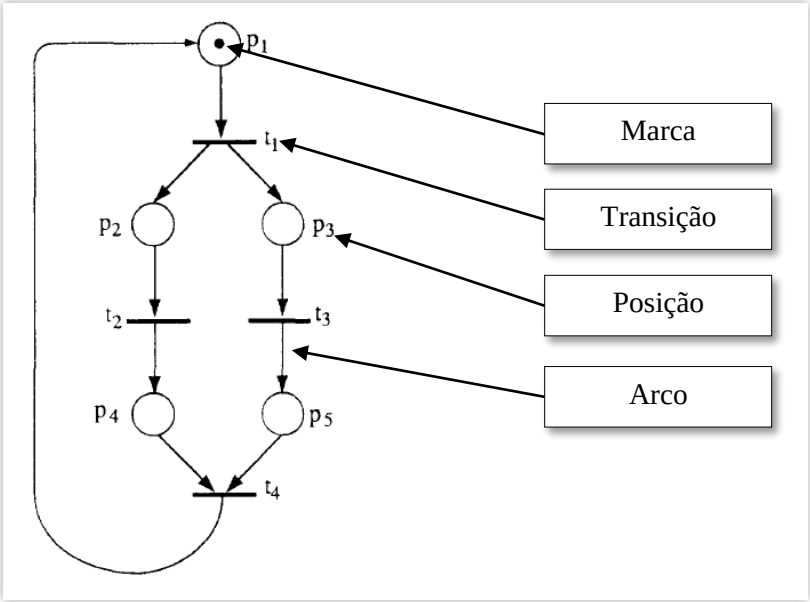


Figura 11 – Exemplo de representação gráfica de uma *Petri Net* [22].

A dinâmica das *Petri Nets* funciona da seguinte forma: para que uma transição esteja habilitada é necessário que todas as posições de entrada da referida transição contenham pelo menos uma marca. Quando ocorre um evento que associado a uma determinada transição, essa condição é

verificada e a Rede de Petri altera o seu estado: é decrementada uma marca da posição de entrada e é incrementada uma marca na posição de saída da transição em causa [11].

A Figura 12 ilustra um exemplo da movimentação de marcas de uma dada posição após a verificação das condições de transição.

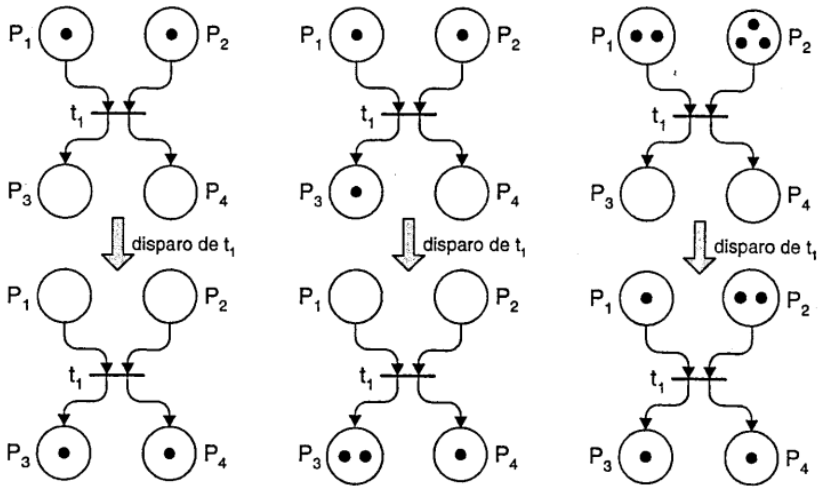


Figura 12 - Movimento de marcas após a verificação das condições de transição [11].

GRAFCET

As *Petri Nets* têm sido tidas como uma alternativa para a descrição de sistemas de controlo. No entanto, apesar das suas vantagens em termos de análise e capacidade de representar a simultaneidade, a perceção industrial era de que esta técnica era muito difícil de usar. Isso levou em 1977 para o desenvolvimento do GRAFCET (*Graphe Fonctionnel de Commande Etape/Transition*), que está intimamente relacionado com um subconjunto de uma *Petri Net* [23].

O *GRAFCET* é por isso, a solução que, por acaso, a *Omron* não disponibiliza, mais utilizada entre os profissionais envolvidos na área da automação. Esta técnica permite descrever, de forma sucinta e objetiva as funções de controlo relativas a um determinado sistema, independentemente do seu campo de aplicação. As principais caraterísticas dos GRAFCETs são a apresentação sintética, potencialidade de representar sistemas complexos, facilidade de interpretação, modelação de sistemas lógicos, metodologia estruturada (de forma descendente) que permite a conceção do geral para o particular, de forma hierarquizada [23].

Os sistemas descritos em GRAFCET são representados através de uma série de etapas, às quais estão associadas ações, sobre as quais evoluem quando se verificam determinadas condições (transições, às quais estão associadas recetividades). As etapas e transições estão ligadas através de arcos, com uma determinada orientação. As ligações orientadas ligam as etapas às transições e as transições às etapas. A aplicação deste formalismo é definida por um conjunto de regras de sintaxe e de evolução descrevendo a dinâmica do sistema a partir de uma situação inicial [12][23].

Usando o GRAFCET torna-se importante conseguir transformar o modelo desenvolvido num conjunto de equações, que sirva de base para a implementação do comando do sistema. Na

Figura 13 é apresentado um exemplo de um GRAFCET tipo e, ao lado, o conjunto de equações correspondentes, que pode servir de base à implementação do respetivo algoritmo de comando [12].

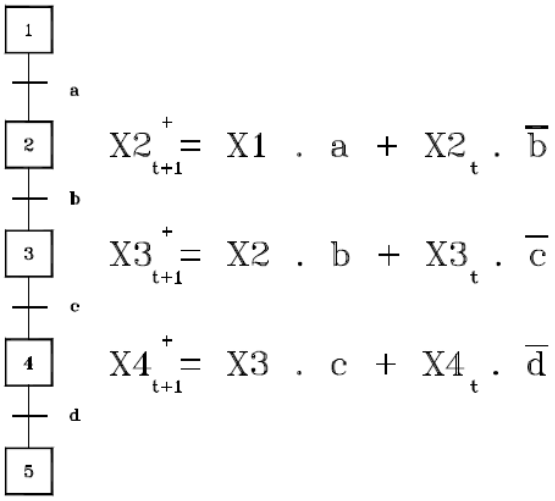


Figura 13 – Exemplo de modelização algébrica de um GRAFCET tipo [12].

Sempre que a descrição de o funcionamento de uma máquina ou parte de máquina se torna complexa, pode ocorrer a situação em o GRAFCET, que descreve o seu funcionamento, se torne de difícil compreensão. Nessa situação, a resolução do problema passa por utilizar macro etapas. Este tipo de representação permite, através de macro representações, simplificar a abordagem ao problema. As macro etapas são também uma forma de facilitar a hierarquização das sequências e dos GRAFCETs, entre eles.

Técnicas de estruturação e plataforma de desenvolvimento do *software*

Aliados aos métodos referidos anteriormente, existem abordagens e técnicas que permitem melhorar a produtividade e qualidade do *software* desenvolvido. Refere-se com isto, a utilização de conceitos como a programação orientada por objetos que permitam acrescentar algumas das caraterísticas referidas na secção 2.3. A norma IEC 61131-3 fornece uma base que permite implementar programas de PLC de acordo com as caraterísticas da abordagem orientada por objetos.

Programação orientada por objetos

A Programação Orientada por Objetos é um tipo de programação popular que utiliza "Objetos" para projetar aplicações e programas para computadores. Um objeto denomina uma estrutura de dados que consiste em campos de dados e métodos de processamento dos dados.

Uma das grandes vantagens da POO é que permite ao programador modelar e desenvolver programas complexos com relativa facilidade, com a vantagem de poder beneficiar de conceitos como a reutilização de componentes, herança, polimorfismo, entre outros.

A solução de um problema visto sobre o ponto de vista do paradigma POO é definir os objetos necessários e definir como eles cooperam entre si e fazer a comunicação entre eles [24].

Este conceito possui algumas caraterísticas interessantes do ponto de vista de desenvolvimento de *software* para PLC, nomeadamente a modularidade, organização e reutilização de *software*.

Norma IEC 61131-3

A norma IEC 61131-3 define um conjunto de linguagens de programação para resolver uma ampla gama de problemas tecnológicos. Outra vantagem é que proporciona diferentes mecanismos para permitir a reutilização de *software*. Esses mecanismos permitem a decomposição do programa principal em módulos independentes definidos por *Program Organization Units* (POUs). Os POUs são divididos nos seguintes elementos que permitem a reutilização de *software*:

Programas: contém a maior parte dos elementos do programa, variáveis. É o corpo do controlo duma máquina ou de um processo.

Function Blocks (FB): são elementos de *software* que podem ser definidos em duas partes: (i) os **dados**, que podem ser parâmetros de entrada/saída, passagem, e as variáveis internas. As variáveis de entrada só podem ser alteradas no exterior e apenas lidas no interior da FB. As variáveis de saída podem ser lidas e alteradas no interior e apenas lidas no exterior das FB. As variáveis de passagem, muitas vezes também designadas por variáveis *In/Out*, são geralmente, estruturas de dados partilhadas com variáveis que podem ser lidas e alteradas no interior das FB. As *Function Blocks* permitem que os valores das variáveis internas fiquem guardados durante a execução dos ciclos de trabalho do autómato; (ii) o **algoritmo** é um código que é executado cada vez que a *Function Block* é evocada. O algoritmo das *Function Blocks* pode ser implementado em linguagem gráfica ou textual, isto é, como por exemplo permite o uso de linguagem *Ladder* ou *Structured Text*. Desta forma o utilizador pode implementar o algoritmo com a linguagem que mais se adeque à exigência da aplicação. Permite também evocar outras *Function Blocks* e *Functions*, resultando então numa forte decomposição hierárquica do programa.

Functions: são elementos de *software* análogos às *Function Blocks*, com a diferença que apenas retomam uma saída e, possuem a particularidade de não memorizarem o estado das variáveis internas. Dessa forma, cada vez que uma *Function* é evocada, desempenha serviços, retomando o programa sempre ao estado inicial [1].

2.4. Modelação da interface homem-máquina

Atualmente, as HMI modernas possuem aplicativos multifuncionais e fornecem ferramentas que permitem a implementação de diferentes modelos de objetos. Estes podem ter diferentes níveis de complexidade, por exemplo, podem ser tratados objetos simples que poderão ser integrados em janelas (botões, caixas de seleção, etc.), como sistemas mais complexos que podem incluir vários tipos de objetos mais simples num objeto só, estes últimos podem mesmo tratar-se de uma janela completa desenvolvida de forma a poder ser reutilizável em vários projetos. Trata-se então de uma abordagem baseada na filosofia de Programação "Orientada por Objetos", mas em ambiente de desenvolvimento de interfaces Homem-máquina. Isto resulta do mesmo propósito que se tem apresentado até este momento, que é a procura da redução de tempo de desenvolvimento do programa para a HMI [25].

3. METODOLOGIA PROPOSTA

Esta secção tem como objetivo descrever o modelo adotado no projeto e desenvolvimento de módulos de controlo implementados em blocos funcionais. Foi necessário realizar o estudo do problema e definir a estratégia. Esta fase do trabalho tomou os seguintes momentos:

- Decomposição estrutural do equipamento;
- Definir um modelo de *software*;
- Desenvolvimento da lógica algorítmica com auxilio a diagramas;
- Criação das estruturas de dados;
- Implementação das soluções;
- Verificação das soluções implementadas.

3.1. Estudo do problema e metodologia proposta

No projeto de equipamentos de montagem de componentes os processos de controlos são tipicamente realizados por recurso a PLC. Para empresas, como a Siroco, dedicadas ao desenvolvimento deste tipo de equipamentos, em que o grau de repetibilidade de projetos que surgem é quase nulo, implica que para cada novo projeto seja necessário desenvolver um novo *software* de controlo. Tipicamente, quando assim se trata, procura-se usar partes de códigos que já tenham sido desenvolvidos e que sejam, de alguma forma, semelhantes ou que possam servir como base ao novo projeto. No entanto, esta prática torna-se complicada, pois a base dos programas desenvolvidos depende de linguagens de programação de baixo nível (*Ladder*) para a implementação de algoritmos de controlo, pois estas linguagens apresentam evidentes limitações quanto se trata da estruturação e modularização do *software*. Além disso, a migração de um *software* desenvolvimento para uma determinada aplicação a partir de uma plataforma para outra é praticamente impossível de executar.

A fim de evitar algumas das desvantagens da técnica de "*copy & paste*" e, para aumentar a qualidade do *software*, foi estudado um conceito que se baseia no desenvolvimento de módulos de controlo independentes e reutilizáveis que, interligados entre si, permitam ao programador implementar um algoritmo de controlo geral, com a vantagem que a configuração possa ser feita de forma simplificada, rápida e automática.

O projeto passa então por estudar vários métodos de programação e procurar seguir a metodologia mais apropriada para a implementação destes módulos. Cada módulo de controlo deverá ser desenvolvido de forma independente e terá o propósito de gerir e controlar as principais estruturas e elementos de um equipamento de montagem de componentes para que quando seja necessário implementar um novo projeto, este seja feito garantindo que:

- Seja de fácil adaptabilidade;
- Responda de forma eficaz;
- Seja reutilizável;

- Possua estruturas que garantam a sua aplicabilidade e versatilidade.

Para projetos de grandes dimensões onde exista a necessidade de reduzir tempo de desenvolvimento e de ter diversas equipas a trabalhar em simultâneo, em diferentes partes do código, a utilização de módulos permite facilitar o desenvolvimento, implementação e manutenção do algoritmo de controlo geral.

Nos últimos anos, a necessidade de novas linguagens e ferramentas para o desenvolvimento de algoritmos em ambiente de programação de autómatos aumentou drasticamente. É por isso que a IEC publicou uma norma para a padronização de linguagens de programação de PLC, denominada de IEC 61131-3. Esta norma introduz princípios da programação de computadores na programação de autómatos, que têm algumas caraterísticas interessantes em termos de modularização e reutilização de *software*. Os mecanismos disponibilizados pela norma, nomeadamente as linguagens de programação e as estruturas (*Functions e Function blocks*), podem ser usadas como potenciais formas para melhorar o desenvolvimento e manutenção de *software* para PLC e servir de base para o desenvolvimento dos módulos propostos.

A análise teórica e os projetos de controlo de equipamentos automatizados têm sido sempre alvos de estudo, principalmente abordando sistemas de eventos discretos e a sua descrição via máquinas de estados finitos ou redes de Petri. Contudo essas abordagens raramente têm sido aplicadas na prática, principalmente porque é difícil derivar um modelo preciso do fabrico de máquinas, que devem incluir modos de funcionamento, alarmes, gestão de operações, entre outros. Para tal, torna-se mais fácil recorrer ao uso de GRAFCETs. Esta técnica permite descrever de forma sucinta, objetiva e hierarquizada todas as funções de controlo relativas a um determinado sistema.

Este projeto focar-se-á também em como introduzir alguns princípios da filosofia da programação orientada por objetos para a programação de controladores industriais (PLC). A abordagem *Object-Oriented* (O-O) tem boa aplicabilidade na modelação de sistemas físicos complexos e com relativa facilidade para o programador, que ficará com o lucro dos benefícios do conceito de reutilização de componentes, entre outros. A Programação Orientada por Objetos possui alguns conceitos que são implementados em diferentes linguagens de programação: objetos, atributos, classes e métodos. Estes conceitos podem ser descritos como sendo sistemas programados, concebidos como um grupo de blocos funcionais, com certos atributos que permitem a comunicação entre si e que reagem de acordo com a implementação dos métodos implementados.

Com a norma IEC 61131-3 passaram a existir estruturas programáveis, tais como a *Function Block* (FB) e *Function* (F). Estes blocos funcionais vêm permitir ao utilizador desenvolver algoritmos de controlo com uma abordagem orientada a objetos. De facto, estas *Function Blocks*, tal como as "classes", são uma definição de ações e atributos. Estas estruturas permitem a implementação de módulos com algoritmos genéricos que, programados corretamente, podem ser usados para implementar objetos de controlo de periféricos e *hardware*. Por exemplo, um bloco funcional para controlo de um cilindro pneumático possui ações, como o avanço ou recuo do cilindro e atributos, como o tempo limite de avanço, ou o estado de avanço/recuo do cilindro. Com a criação de um único bloco funcional para o controlo de um cilindro pneumático, ficam definidas neste mesmo bloco, todas as ações e atributos para cada cilindro pneumático semelhante ao utilizado no sistema. Estes blocos funcionais podem ser desenvolvidos de forma independente do programa principal, possibilitando

então a criação de uma biblioteca de blocos funcionais e, desta forma, facilitar a manutenção e reutilização dos mesmos.

Em termos práticos, o desenvolvimento de objetos obriga a decompor a estrutura de uma máquina em partes mais pequenas (Estações de trabalho, robôs "*pick-and-place*", etc.), que depois de identificadas e isoladas, algumas podem ainda ser decompostas outra vez em mais objetos, cooperando assim de forma hierárquica, em que cada uma delas estará associada a uma ação ou conjunto de ações particulares. Isto é, a maneira mais eficaz e eficiente passa por partir do problema geral para a divisão em blocos cada vez mais pequenos e mais fáceis de lidar. Assim, os blocos encontram a solução por eles próprios ou entrando em contato com outros blocos, sendo capazes de resolver problemas específicos. Isto é o que é chamado de abordagem *Top-Down*, ver Figura 14.

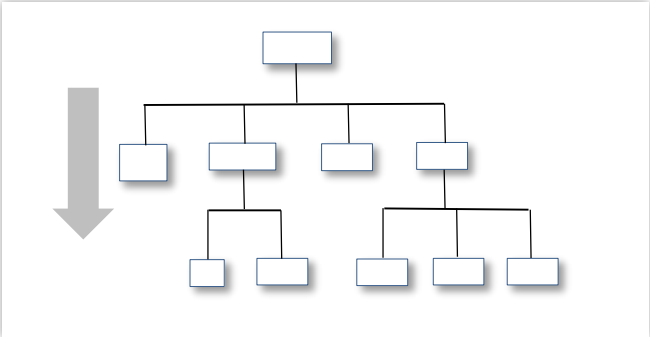


Figura 14 - Abordagem Top-Down na Implementação de FB.

Geralmente, a decomposição estrutural de *hardware* para a definição de objetos trata-se da primeira fase do trabalho. Por imposição da proposta do projeto, a divisão modular foi feita, tendo em conta as partes mais relevantes para o conceito de "equipamento standard", resultando então que as estruturas mais relevantes deviam tomar por base os módulos principais. A restante estruturação foi feita seguindo os mesmos princípios seguidos.

Uma vez definida a estrutura dos objetos, os seus comportamentos devem ser analisados e descritos. Recordando o paralelismo com a Programação Orientada por Objetos, os algoritmos de controlo devem ser vistos como métodos que operam na parte física do objeto (sensores, atuadores), memorizando o estado físico (posição, operações executadas, etc.) em variáveis que armazenam essa informação em posições específicas da memória. É evidente que os componentes de *software* do objeto (algoritmos e dados) devem ser isolados do resto do programa e encapsulados em estruturas de *software* para que, caso seja necessário, utilizar novamente um objeto do mesmo tipo com caraterísticas similares, em qualquer outra parte do programa, ou mesmo, noutro projeto, o bloco de *software* possa ser facilmente reutilizado.

Estruturas de dados

De acordo com a norma IEC 61131-3, são definidos dados elementares. Porém, existem mecanismos que permitem ao utilizador criar os seus próprios tipos de dados, denominados de "*user data type*" (UDT).

A norma não define tipos de dados específicos para os UDT. O objetivo das UDT é permitir ao utilizador criar as suas próprias estruturas de dados (*Structures*). Estas estruturas de dados são *data types* que contêm mais elementos que não têm de ser do mesmo tipo de dados. Uma estrutura

de dados pode ser implementada de forma hierárquica, o que significa que uma estrutura pode ser um elemento de uma outra estrutura. As estruturas de dados e os *arrays* podem ser combinados, isto é, um *array* pode ser um elemento de uma estrutura de dados, sendo também possível criar um *array* de tipo UDT criado pelo utilizador, ficando então a ser um *array* cujos elementos são estruturas de dados (ex. ARRAY[0..N] OF UserDataType) e que pode ter uma ou mais dimensões dependendo da complexidade da estrututa.

Elaboração da interface HMI

Na elaboração de um projeto, o desenvolvimento da interface Homem-Máquina é um processo demorado tal como a elaboração do algoritmo do controlador. Contudo, na elaboração de janelas de interface, onde o grau de repetibilidade de algumas janelas pode ser elevado, torna-se um processo repetitivo e tedioso em que, mais uma vez, se recorre muitas vezes ao método de "*Copy & Paste*". Para evitar o mesmo tipo de desvantagens que surgem no desenvolvimento de código para os PLCs, procura-se adotar métodos e ferramentas que permitam desenvolver e implementar objetos e janelas padrão reutilizáveis.

Estes objetos podem conter uma mistura de gráficos, objetos pré-definidos, animações, variáveis de entrada e saída e até algoritmos de controlo, e posteriormente adicionados à *toolbox* do projeto da interface. Estes objetos são implementados e aplicados com os mesmos princípios dos blocos funcionais, ou seja, funcionam com métodos encapsulados que executam funções. Consequência disso, estes objetos são caraterizados por serem facilmente reutilizáveis e garantirem eficiência da sua aplicabilidade, permitindo assim reduzir o tempo de desenvolvimento das interfaces Homem-Máquina.

3.2. Implementação das metodologias propostas

Esta secção destina-se a descrever a implementação das metodologias definidas para o desenvolvimento dos algoritmos de controlo dos blocos funcionais, e do desenvolvimento das janelas de interface padrão. Para cada caso desenvolvido, é feita uma breve descrição de como estes podem ser introduzidos nos programas de PLC.

3.2.1. Desenvolvimento dos algoritmos de controlo

A metodologia descrita na secção anterior foi aplicada no desenvolvimento dos blocos funcionais propostos pela empresa Siroco: algoritmos de controlo para cilindros pneumáticos, pratos rotativos e estações de trabalho. O esquema da Figura 15, ilustra a sequência de implementação do método para a elaboração destas estruturas.

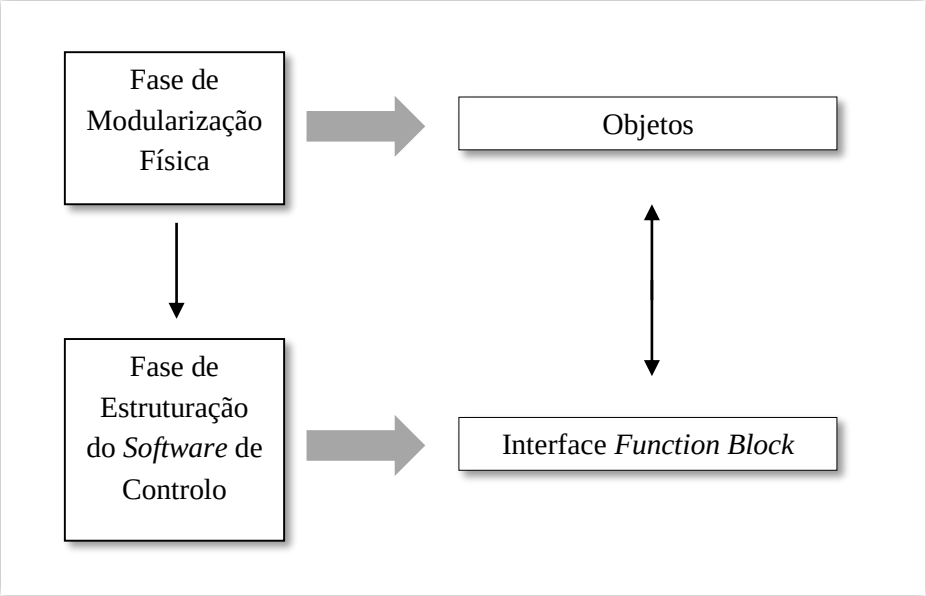


Figura 15 - Esquema da metodologia proposta.

Na primeira fase de trabalho, começa-se por definir a modularização física da estrutura do equipamento. A primeira parte da modularização foi definida pelos responsáveis da automação do departamento de desenvolvimento da empresa Siroco e, a partir daí, fez-se a restante decomposição destas estruturas principais, em componente mais específicos. Finalizada a primeira etapa, inicia-se a fase de estruturação do *software* de controlo em que, para cada objeto principal, é desenvolvido uma interface *Function Block* na qual, serão definidas propriedades e métodos intrínsecos a esses objetos.

Dado que a empresa Siroco utiliza correntemente o pacote de *software CX-One* da *Omron*, e pretende atualizar-se para a versão mais recente, foi proposto que estes blocos funcionais fossem desenvolvidos no *Sysmac Studio*, a pensar já em utilizações futuras.

O *Sysmac Studio* é a última versão do *software* de programação de autómatos da *Omron*. Este *software* permite ao utilizador criar, configurar, programar e simular um conjunto de dispositivos, como, por exemplo PLCs e HMIs, utilizando apenas um pacote de *software*. Deste modo, a complexidade da configuração é significativamente reduzida permitindo assim mais versatilidade na programação ou configuração de sistemas de automação.

O *Sysmac Studio* oferece ainda uma plataforma de *software* para todos os tipos de controladores da Omron. Isto permite uma fácil conversão e reutilização de código entre diferentes modelos de PLCs. O ambiente de programação do controlador possui a potencialidade de desenvolver blocos funcionais que podem ser programados em *Ladder* ou *structured text*, permitindo assim adequar e aproveitar as potencialidades de cada linguagem à aplicação desenvolvida.

O *Sysmac Studio* define um modelo para programação modular nos seus PLCs. Esta ferramenta é compatível com a norma IEC 61131-3 e permite gerar blocos de algoritmos encapsulados genéricos e reutilizáveis denominados de *Functions* (F) e *Function blocks* (FB). O *Sysmac Studio* permite ao utilizador criar as suas próprias estruturas de dados (*Structures*). Estas estruturas de dados são *data types* que possuem as propriedades que definem os módulos aos quais

estão associados. A estrutura de dados deve garantir a aplicabilidade e versatilidade do objeto. Para tal, deve prever o máximo de propriedades e características que o objeto pode ter, para que a sua versatilidade e aplicabilidade seja maior.

Para se aceder aos elementos de uma estrutura de dados deve-se escrever o nome da estrutura de dados, no fim escrever um "ponto" e, de seguida, escrever o nome do elemento. O formato será por exemplo: "variableName.elementName". Caso se trate de um *array* do tipo UDT, o formato será semelhante mas é necessário identificar o elemento do array que queremos aceder, por exemplo: "variableName[i].elementName". O mesmo acontece para uma estrutura de dados cujo elemento se trata de um *array*, por exemplo: "variableName.elementName[i]".

Os blocos funcionais serão o foco central neste projeto. A funcionalidade destes blocos será controlar estruturas especificas de um equipamento de montagem de componentes.

O desenvolvimento das soluções padrão modulares na automatização de sistemas de montagem de componentes, deve ser feito de forma a tornar possível a sua reutilização e integração em algoritmos mais complexos. A elaboração destes blocos funcionais deve começar pelos componentes mais simples da estrutura até aos mais complexos. Isto é, partindo do pressuposto que tipicamente um equipamento de montagem de componentes é composto por uma linha de montagem com um tapete rolante que transporta peças, colocadas em paletes, que percorrem uma série de estações de trabalho, em que cada uma destas estações pode executar o controlo e monitorização das operações executadas através de atuadores e sensores específicos.

A primeira fase de elaboração destes blocos, passará pela identificação dos elementos mais simples, que neste caso são os atuadores e sensores alocados nas estações de trabalho. Uma vez identificados, estes elementos deverão ser alvo de estudo e análise. Esta é uma etapa importante para compreender de que forma podem ser controlados e monitorizados, pois na estruturação dos dados do componente deve-se antever diferentes configurações e especificidades destes elementos. Por exemplo, no caso dos atuadores pneumáticos, estes atuadores podem ser de vários tipos, rotativos ou lineares, de simples ou duplo efeito, podem ser controlados por diferentes tipos de válvulas e pode variar o número de sensores inseridos na monitorização do estado do atuador, etc. Resumindo, uma estrutura de dados de um módulo de controlo possui membros associados às funções gerais e deve estar associada ao bloco funcional respetivo, não esquecendo que devem ser incluídos elementos que permitam comunicar com os restantes blocos que englobam o sistema. Por isso, a fase da estruturação adquire mais importância quanto mais quisermos desenvolver um módulo que seja robusto e capaz de garantir um elevado grau de aplicabilidade e versatilidade. Uma vez criada a estrutura de dados, dá-se início à implementação do programa geral.

O mesmo se passa com os restantes elementos estruturais do equipamento; deve-se seguir o procedimento de forma hierárquica, evoluindo consoante o grau de complexidade.

3.2.2. Desenvolvimento da interface Homem-Máquina

O *Sysmac Studio* permite a criação de ecrãs para um projeto de HMI. Este *software* possui ainda a particularidade de não ser necessário seguir o mapa de memória interna do controlador. Isto é, o *Sysmac Studio* atribui automaticamente um endereço de memória às variáveis criadas pelo utilizador, esta autogestão dos endereços das variáveis promove um desenvolvimento mais rápido e reduz erros. Isso também evita a necessidade de esperar por definições de endereçamento de memória

de *hardware* antes do início de desenvolvimento do *software*. Desta forma, permite criar projetos que não são dependentes de dispositivos ou mapas de memória específicos e que podem ser projetados de forma independente e desenvolvidas em simultâneo com os programas do controlador.

O *Sysmac Studio* define ainda um modelo para a implementação de objetos reutilizáveis, denominado de IAG (*Intelligent Application Gadget*). Os IAGs contêm uma mistura de animações, variáveis e métodos de código VB (*Visual Basic*). Um projeto IAG pode ter o seu próprio código VB que se utiliza para realizar alguma funcionalidade prevista. Os IAGs podem permitir que o seu código VB possa ser chamado desde o projeto que o contém. Porém, são "caixas negras", semelhante a um FB do PLC, mas para HMI. A Figura 16 e Figura 17 ilustram um exemplo de um IAG e o seu código, respetivamente. Este IAG trata-se de um botão que pode ser adicionado em qualquer janela de um projeto e a sua função é sinalizar através de cores a ocorrência de um alarme (verde – sem ocorrências; amarelo – alarme detetado), permitindo ainda aceder à janela de monitorização de alarmes.

Figura 16 - IAG "Botão de Alarmes".

```
Sub Main
    If (VAR_HMI_AlarmsRaised>0) Then
        Me.Rectangle0.IsVisible=False    'Verde
        Me.Rectangle1.IsVisible=Not(Me.Rectangle1.IsVisible)
        system.Threading.Thread.Sleep(150)
    Else
        Me.Rectangle0.IsVisible=True    'Amarelo
        Me.Rectangle1.IsVisible=False
    End If
End Sub

Sub MostrarPagina

    ShowPage(AlarmPage)

End Sub
```

Figura 17 - Algoritmo de controlo do IAG "Botão de Alarmes" (linguagem *Visual Basic*).

Múltiplos IAGs podem estar compactados num único arquivo IAG, sendo conhecido como uma coleção de IAGs. O arquivo de IAG contém os nomes e categorias de cada IAG, sendo estes impossíveis de modificar pelo utilizador final do IAG.

Uma vez instalados e inseridos na caixa de ferramentas de um projeto, os IAGs comportam-se como outros objetos da caixa de ferramentas, podendo ser arrastados para a página em que serão usados.

O comportamento em *runtime* do IAG é controlado por variáveis do projeto. Estas variáveis são acedidas a partir das propriedades do IAG. A variável pode ser usada para extrair dados do IAG ou para proporcionar dados ou parâmetros para mudar o comportamento.

A elaboração de objetos padrão, através dos modelos definidos pelos IAG, torna-se fundamental neste projeto pois permitirá a elaboração de janelas padrão. Estas janelas padrão têm o objetivo de tornar a implementação da interface para um determinado projeto um processo mais rápido. Com recurso a objetos facilmente parametrizados, a elaboração das janelas mais utilizadas em projetos passará a ser mais fácil. Alguns exemplos de janelas padrão são a janela de monitorização e controlo dos cilindros pneumáticos e das estações de trabalho. Esta lista de IAGs inclui ainda um IAG desenvolvido para monitorizar cilindros pneumáticos. Este IAG é um objeto gráfico que permite ao utilizador visualizar o estado do cilindro (avançado/recuado) e o estado dos sensores (ativo/desativo), ver Figura 18. Este IAG de monitorização do cilindro pneumático deve ser inserido na janela dos cilindros pneumáticos, contudo, poderá também fazer parte da janela de controlo e monitorização das estações, nomeadamente na área disposta para o *layout* da estação. Desta forma possibilita um acesso mais direto e simultâneo dos cilindros que compõem a estação.

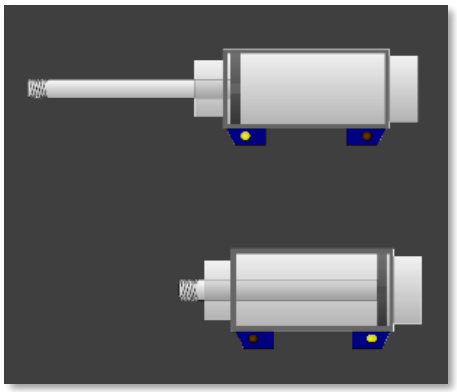


Figura 18 - IAG de monitorização dos cilindros pneumáticos.

Esta metodologia permitirá que, na elaboração de um projeto, se "arrastem" os IAGs que definem as janelas padrão para o projeto e apenas seja necessário introduzir as variáveis do projeto para proporcionar os dados e os parâmetros necessários. A parametrização é feita da seguinte forma ilustrada na Figura 19.

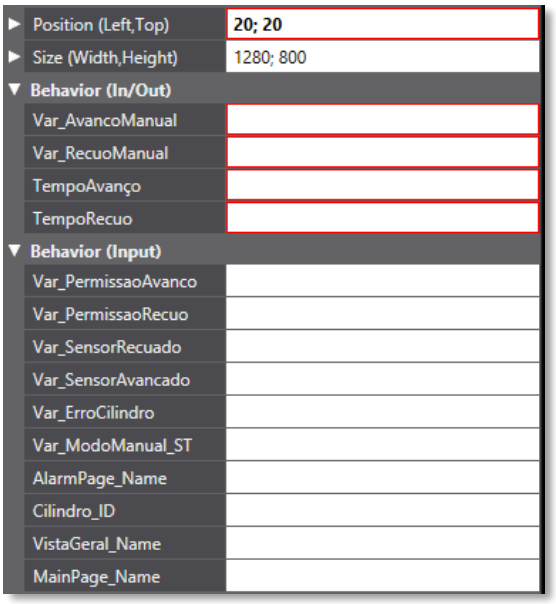


Figura 19 - Exemplo de parametrização de IAG.

3.3. Descrição dos algoritmos de controlo

O programa base de controlo modular é constituído por módulos de algoritmos específicos de controlo. Os módulos principais são os que controlam as estruturas principais do equipamento. Existem, portanto, três algoritmos principais de controlo, em que, cada um pode ter blocos funcionais adjacentes mais específicos detalhando desta forma a arquitetura do sistema de controlo, o que permite uma implementação mais eficaz. De seguida apresenta-se uma descrição geral do funcionamento e aplicação dos blocos funcionais.

3.3.1. Controlo das estações de trabalho

As estações de trabalho estão inseridas em linhas de montagem e o número de estações numa linha varia de máquina para máquina. Cada estação de trabalho desempenha uma função especifica na linha, mas o controlo é comum. Assim, para cada estação que a linha possua, é necessário adicionar um módulo de controlo e monitorização. Uma estação de trabalho pode ser, por exemplo, uma estação inserida numa linha de montagem de placas PCB, em que a sua função é inserir componentes eletrónicos nas placas que chegam à área de trabalho dessa mesma estação.

Programa geral:

O módulo de controlo e monitorização desta estrutura é composto por um bloco funcional que tem como principal funcionalidade estabelecer o modo de operação das estações e da máquina. O algoritmo desenvolvido neste bloco funcional possui uma estruturada de dados específica, partilhada com a HMI, para permitir ao utilizador alterar o modo de funcionamento da máquina. As estações de trabalho podem funcionar nos seguintes modos de funcionamento:

- Automático;
- Manual;
- Reposição;

- *Bypass*.

Dentro do modo automático, existem ainda os comandos específicos que definem o modo em que a estação vai funcionar. Alguns dos comandos específicos mais importantes são:

- Produção;
- Fim de Produção;
- Simulação;
- Passo a passo.

Estes comandos específicos podem ser adicionados consoante a aplicação e requisitos do cliente. Em todas as máquinas com mais do que uma estação e que processam a mesma peça deve existir uma estação *Master* de controlo geral da máquina. Cada estação tem os seus próprios comandos de controlo individuais. Porém, os comandos da estação *Master* devem estabelecer o modo de funcionamento geral da máquina.

A estrutura de dados deste módulo é composta por membros que servem essencialmente para comunicar com os elementos constituintes da estação de trabalho, como por exemplo, Atuadores, Sensores, entre outros. Esta informação vai condicionar o modo como estes elementos vão desempenhar as suas funções.

Foi ainda necessário implementar uma *Function Block* para controlar o funcionamento da linha de produção e das estações (Figura 21). O algoritmo geral de funcionamento deste bloco funcional está descrito no GRAFCET apresentado no **Anexo 1**. Em linhas de montagem deste tipo, em que o funcionamento é assíncrono, o tapete rolante encontra-se em movimento contínuo, visto que serve para transportar as paletes de estação em estação. Contudo, devem existir sistemas de travagem das paletes, os denominados "*Stoppers*", estes dispositivos são comandados para controlar o fluxo das paletes. Ver Figura 20.

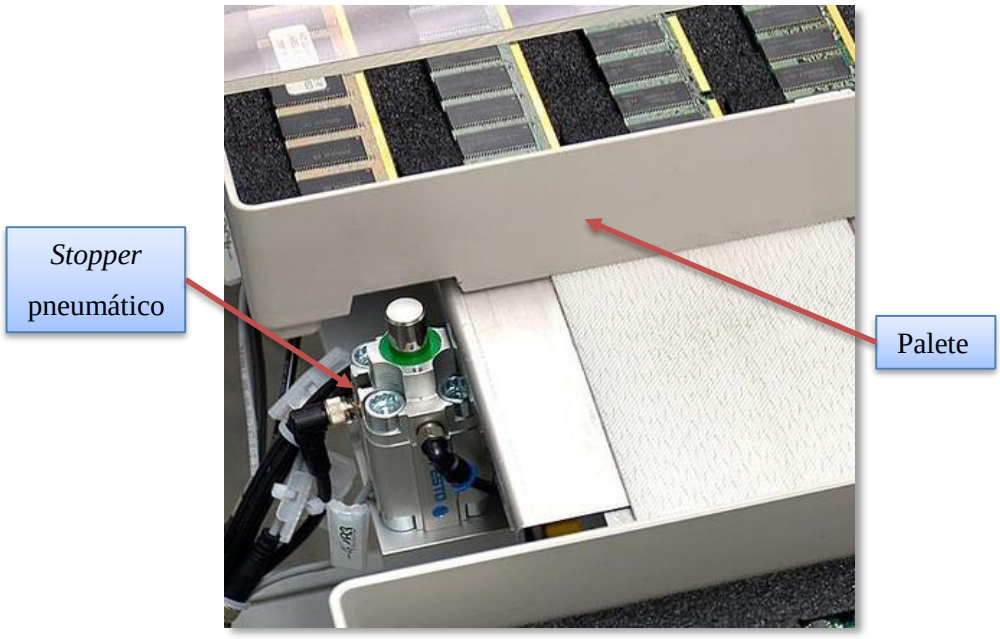


Figura 20 – *Stopper* pneumático (*Festo DFSP*) numa linha de montagem – [26].

O propósito desta *Function Block* é de controlar o sincronismo da linha, e para tal, o algoritmo geral controla o funcionamento dos *Stoppers* de entrada e saída da estação.

Para controlar o fluxo das paletes o algoritmo descrito nesta *Function Block* controla o funcionamento dos *Stoppers* de entrada da estação "atual" e de saída da estação anterior. Ou seja, o *Stopper* de saída da estação anterior, tem a função de travar a palete, caso a próxima estação para a qual a palete se dirige se encontrar indisponível, seja por estar a executar algum processo ou por algum outro motivo, deixando desta forma a peça em estado de "espera". A estação "anterior" é identificada na FB como "Estação 1" e a estação seguinte é a "Estação 2".

Este módulo controla ainda as ações das estações, ou seja, em função do modo de funcionamento e da presença de peça dá, ou não, o comando de "início de operação". Mais especificamente, o algoritmo deste bloco funcional adota determinadas ações com base no modo de funcionamento e os comandos específicos definidos para a estação de trabalho. Por exemplo, se a máquina estiver em modo de "produção" e uma das estações estiver em modo "bypass", a estação permanecerá em estado de "não operação" e qualquer peça que passe por essa estação fica em espera até que a estação seguinte esteja disponível para poder avançar. Esse controlo é aplicado à "Estação 2".

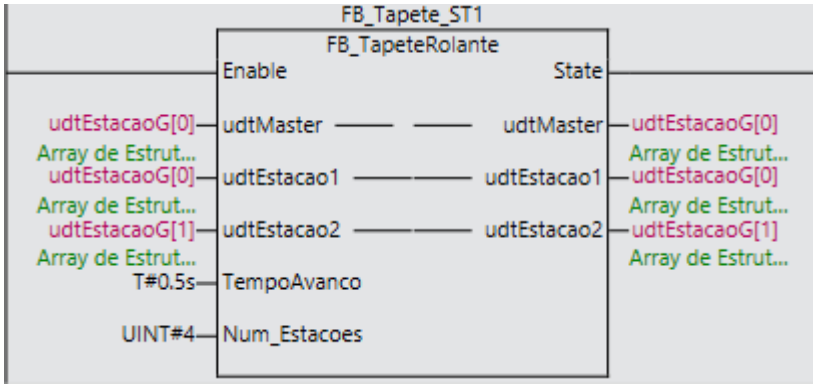


Figura 21 – *Function Block* de controlo do funcionamento da linha de produção e estações.

Dentro da *Function Block* de controlo do funcionamento da linha de produção, existe ainda uma *Function* para o controlo do "fim de produção" em modo automático, da máquina (ver Figura 22). Esta função funciona da seguinte maneira: Quando o operador aciona o comando de "fim de produção", o algoritmo começa por retirar de serviço o posto de abastecimento, impedindo que entrem mais peças na máquina. Depois, a última peça colocada na máquina será a referência para desativar as estações de trabalho à medida que a peça percorre a linha de montagem. Isto é, quando a estação, onde a peça se encontra terminar o processo, é colocada em modo "fim de produção". Quando todas as estações estiverem em "fim de produção", terminou o processo.

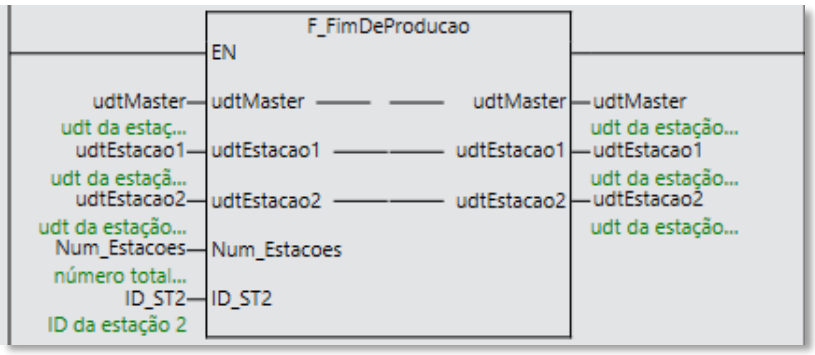


Figura 22 – *Function* de controlo do "fim de produção" das estações.

Descrição dos processos:

Nesta secção serão apresentados os processos de seleção dos modos de funcionamento da estação de trabalho.

Processo de colocação da estação de trabalho em modo de funcionamento automático

O GRAFCET da Figura 23, descreve a sequência de procedimentos que devem ser efetuados para colocar a estação no modo de funcionamento automático. Neste GRAFCET o programa está implementado da seguinte forma:

Etapa 0: o programa principal fica em espera até receber o sinal do botão da HMI para iniciar o modo de funcionamento automático. De seguida é feita a verificação das condições para iniciar o modo de funcionamento automático. Caso as condições iniciais sejam verdadeiras, o GRAFCET avança para a etapa 5. Caso contrário, o GRAFCET avança para a etapa 2.

Etapa 2: nesta etapa, o processo inicia o modo de funcionamento manual da estação. A colocação da estação em modo de funcionamento manual é uma medida de segurança, isto é, com a colocação da estação neste modo, garante-se que todas as etapas do modo automático são desativadas, para de seguida colocar a estação no estado inicial.

Etapa 3: após verificação do modo de funcionamento manual da estação, dá-se a passagem para a etapa 3. Nesta etapa a estação é colocada em modo de "reposição". Esta ação inicia a colocação dos periféricos da estação, nas suas condições de início de trabalho. Com a verificação do modo de reposição da estação de trabalho, ativa a etapa 4. Nesta etapa o programa fica em estado "espera".

Etapa 5: a ativação da etapa 5, é dada com a verificação das condições iniciais da estação. Esta etapa ativa o modo de funcionamento "automático" e termina o processo de colocação da estação no modo de funcionamento automático.

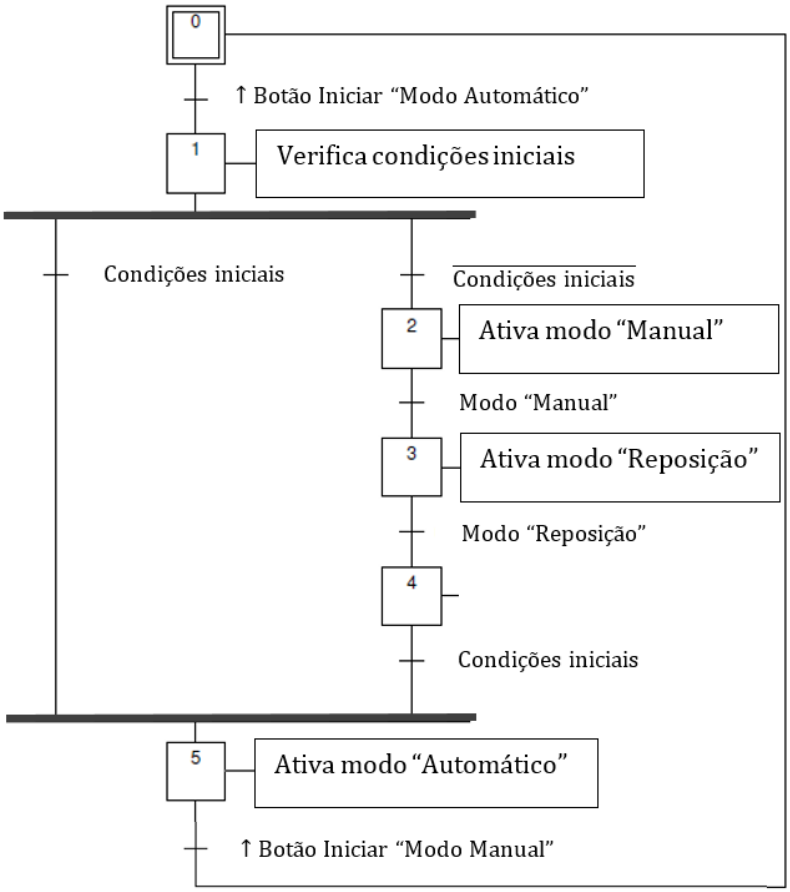


Figura 23 - GRAFCET de colocação da estação em "modo automático".

O GRAFCET anteriormente apresentado descreve o processo de colocação de uma estação em modo de funcionamento automático. Caso o operador pretenda colocar toda a máquina em modo automático, isto é, colocar todas as estações em modo automático, o procedimento repete-se. Com a diferença que, muda o processo de verificação das condições iniciais para o início do modo de funcionamento. Desta vez, o processo de verificação é mais complexo, pois só acontece quando todas as estações verificarem as suas condições iniciais individuais.

Para fazer a verificação das condições iniciais das estações foi criado um *array*, em que cada índice do *array*, corresponde à identificação da estação de trabalho (por exemplo: o índice "1" corresponde à estação "1"). Neste *array* cada estação guarda a informação de que já concluiu o processo de reposição e a condição inicial é verdadeira.

A verificação dos dados guardados no *array*, é feita através do algoritmo apresentado na Figura 24. O processo de verificação do *array* é feito com um ciclo *for* que percorre todas as posições do *array*, e contabiliza o número de condições verdadeiras. Caso o número de condições verdadeiras seja igual ao número de estações de trabalho, estão validadas as condições inicias da máquina.

```
2  //VERIRIFACAÇÃO DAS CONDIÇÕES INICIAS DA MÁQUINA
3  Size:=SizeOfAry(udtEstacaoG)-1; // Size a subtrair por um porque o indice zero é a estação Master
4  Cond_Iniciais:=0;
5  FOR i:=1 TO N_Estacoes BY 1 DO
6      IF (udtMaster.Master.CondInit_Ar[i]) THEN
7          Cond_Iniciais:=Cond_Iniciais+1;
8      ELSE
9          EXIT;
10     END_IF;
11 END_FOR;
12 IF (Cond_Iniciais=N_Estacoes) THEN
13     udtMaster.Master.CondInit:=TRUE;
14     udtMaster.Master.CondInit_Ar[0]:=TRUE;
15 ELSE
16     udtMaster.Master.CondInit:=FALSE;
17     udtMaster.Master.CondInit_Ar[0]:=FALSE;
18 END_IF;
```

Figura 24 – Algoritmo de verificação das condições iniciais da máquina (linguagem *Structured Text*).

Processo de seleção dos comandos específicos da estação de trabalho

O operador deve selecionar o "comando específico", após a colocação da estação em modo de funcionamento automático. Essa seleção deve ser feita com base no modo de operação que pretende que a estação desempenhe. Por exemplo, se pretender fazer uma simulação do funcionamento da estação deve colocar a estação em modo de funcionamento automático e selecionar o comando "simulação".

Processo de colocação da estação de trabalho em modo de funcionamento manual

O modo de funcionamento manual não requer que a estação retorne ao estado inicial. Ao colocar neste modo apenas são desativadas todas as etapas de funcionamento. A ativação deste modo de funcionamento é feita pelo operador, no momento em que pressiona o botão de ativação do modo de funcionamento manual da estação em causa.

Processo de colocação da estação de trabalho em modo de funcionamento reposição

Se o operador pretender repor as condições iniciais da estação, deve começar por colocar a estação em modo manual e, de seguida, pressionar o botão dedicado à ativação do modo de "reposição".

Processo de colocação da estação de trabalho em modo de funcionamento "*bypass*"

Para colocar a estação em modo "*bypass*", o operador deve tomar o seguinte procedimento:

- Colocar a estação em modo manual;
- Colocar a estação em modo de reposição.

E após a verificação das condições iniciais da estação, o operador deve então pressionar o botão de ativação do modo "*bypass*".

Processo de colocação da estação em "fim de produção"

Existe uma *Function* de controlo de "fim de produção" para cada estação. O processo de colocação da estação em "fim de produção" é feito em simultâneo em todas as estações e, quando todas as estações estiverem em modo "fim de produção", significa que já não se encontram mais peças na linha.

O algoritmo de colocação da estação em "fim de produção" (descrito no fluxograma da Figura 25) é apresentado nos seguintes passos:

Início: após a ativação do comando de "fim de produção" da máquina, dá-se o início do processo de colocação "fim de produção" das estações.

E1: o primeiro passo após a ativação do comando de "fim de produção" é desativar a estação de abastecimento da máquina, para impedir a entrada de peças.

C1: a primeira condição verifica se a estação atual se encontra em "fim de ciclo" (isto é, se terminou a tarefa a desempenhar na peça). Caso seja falso, espera que seja concluído o ciclo de trabalho da estação. Caso contrário segue para a condição "C2".

C2: após o "fim de ciclo" da estação atual, o processo vai verificar se há alguma peça nova para processar. Caso exista uma peça a dirigir-se para a estação atual, o processo retorna à condição "C1". Caso contrário segue para a condição seguinte.

C3: esta condição verifica se a estação anterior se encontra em modo "fim de produção". Caso seja verdade, significa que não existem mais peças para a estação atual processar, e, portanto, reúne as condições para entrar em modo "fim de produção". Caso contrário, ainda tem pelo menos mais um ciclo de operação.

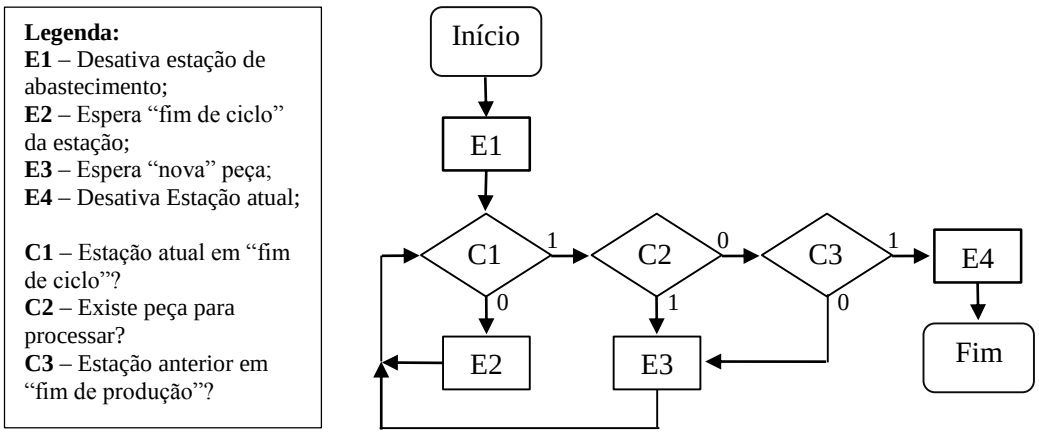


Figura 25 - Fluxograma do processo de colocação de "fim de produção" da estação.

O processo de verificação do "fim de produção" da máquina é feito pelo algoritmo apresentado na Figura 26. Quando as estações entram em modo "fim de produção", cada estação guarda essa informação num *array* que regista a informação de todas as estações. O processo de verificação do *array* é feito com um ciclo *for* que percorre todas as posições do *array*, e contabiliza o número de condições verdadeiras. Caso o número de condições verdadeiras seja igual ao número de estações de trabalho, significa que todas as estações já terminaram a produção, e que não há mais peças para processar, e neste caso a máquina terminou a produção.

```
Cond_FimProd:=0;
FOR i:=1 TO Num_Estacoes  BY 1 DO
    IF (udtMaster.Auxiliares.FimDeProducao_Ar[i]=TRUE) THEN
        Cond_FimProd:=Cond_FimProd+1;
    ELSE
        EXIT;
    END_IF;
END_FOR;

IF (Cond_FimProd=Num_Estacoes) THEN
    udtMaster.Comandos.FimProducao:=FALSE;
    FOR i:=1 TO Num_Estacoes BY 1 DO
        udtMaster.Auxiliares.FimDeProducao_Ar[i]:=FALSE;
        udtEstacao2.Comandos.FimProducao:=FALSE;
    END_FOR;
END_IF;
```

Figura 26 - Processo de verificação do "fim de produção" da máquina (linguagem *Structured Text*).

Aplicação do módulo:

Resumindo, numa linha de montagem com um determinado número de estações, insere-se um bloco por cada estação de trabalho e mais um para o controlo da estação *Master*. A estrutura de dados referente à estação de trabalho ao ser adicionada nas variáveis globais do programa, deve ser declarada como sendo um *array* cujo número de elementos é igual ao número de estações do equipamento, ver exemplo da Figura 27. Desta forma, aceder aos dados de uma estação é uma tarefa simples em que apenas será necessário alterar o índice do *array*, para identificar a estação. Na Figura 28 apresenta-se um exemplo onde é feita a atribuição da "identificação" da *Function Block* que vai controlar a "estação de trabalho 1". Essa atribuição é feita através do índice do *array* da estrutura de dados. A estrutura de dados com o índice "0" é referente à estação "Master" (ou de "controlo geral").

Name	Data Type	Initial Value	AT	Retain	Constant	Network Publish	Comment
udtEstacaoG	ARRAY[0..5] OF UDT_Estacao			☑	☐	Publish Only	Array de Estrutura Estação

Figura 27 - Declaração da estrutura de dados da estação de trabalho (*Sysmac Studio*).

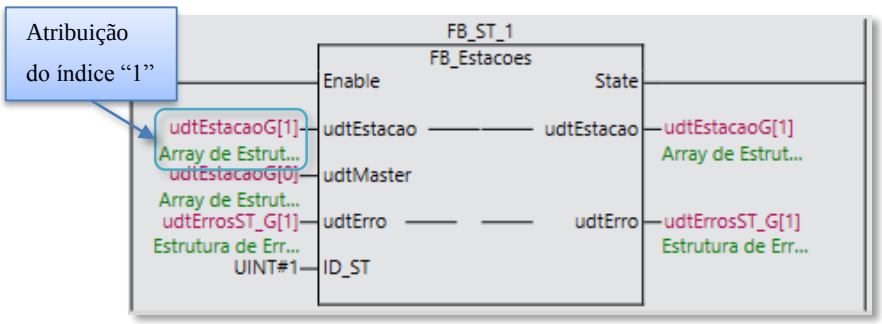


Figura 28 - *Function Block* de controlo da estação de trabalho.

Interface de comunicação com o utilizador:

Por cada estação (incluindo a estação *Master*) existe uma janela na HMI, que permite ao utilizador selecionar os modos de funcionamento e comandos específicos da estação, ver Figura 29. Existe ainda outra janela que possibilita ao utilizador visualizar dados estatísticos, o *layout* físico da estação, e permite aceder à janela de monitorização e controlo dos cilindros dessa estação. Como foi referido anteriormente existe também uma janela de controlo geral da máquina.

Figura 29 - Seleção do Modo de Funcionamento de uma estação de trabalho.

3.3.2. Controlo dos cilindros pneumáticos

Os cilindros pneumáticos (ver Figura 30) são os atuadores aos quais a empresa Siroco mais recorre para inserir nas estações de trabalho, pois são um dos elementos chave de atuação nas máquinas. Uma máquina desenvolvida na Siroco é, por norma, constituída por um número significativo de equipamentos deste tipo, compostos sempre por dois sensores magnéticos, um para indicar se o cilindro se encontra avançado e outro recuado. Uma vez que a sua utilização é feita em grande número, na implementação do algoritmo geral da máquina, a utilização de blocos funcionais dedicados ao controlo dos cilindros pneumáticos será uma mais valia, quando se trata de redução de tempo de desenvolvimento. A empresa considerou então que será importante implementar um módulo de controlo para estes atuadores.

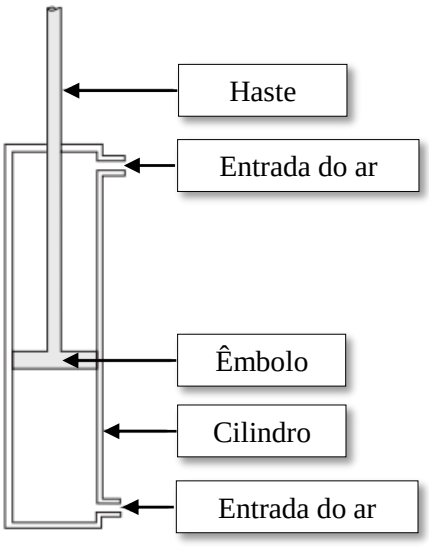


Figura 30 - Cilindro pneumático (duplo efeito) [4].

Programa geral:

Este tipo de atuadores opera nas estações de trabalho, e o facto destes dipositivos terem diferentes modos de operação para os diferentes modos de funcionamento da máquina/estação de trabalho, isto é, podem trabalhar em modo automático, manual e reposição (ver Figura 31). Este facto faz com que seja necessário que o algoritmo deste módulo faça uma verificação cíclica do modo de funcionamento da estação, antes de realizar as operações de avanço e recuo. Esta verificação é necessária e, para tal, a *Function Block* de controlo dos cilindros pneumáticos deve ter acesso à estrutura de dados de controlo das estações de trabalho.

Para qualquer um destes modos de funcionamento existe uma permissão de avanço e de recuo do cilindro (ver Figura 31). O comando de avanço ou de recuo pode vir de qualquer parte do programa desde que tenha acesso à estrutura de dados do cilindro pneumático e, se assim for, que se confirme a permissão do movimento pretendido. Caso contrário, o cilindro não efetua qualquer movimento.

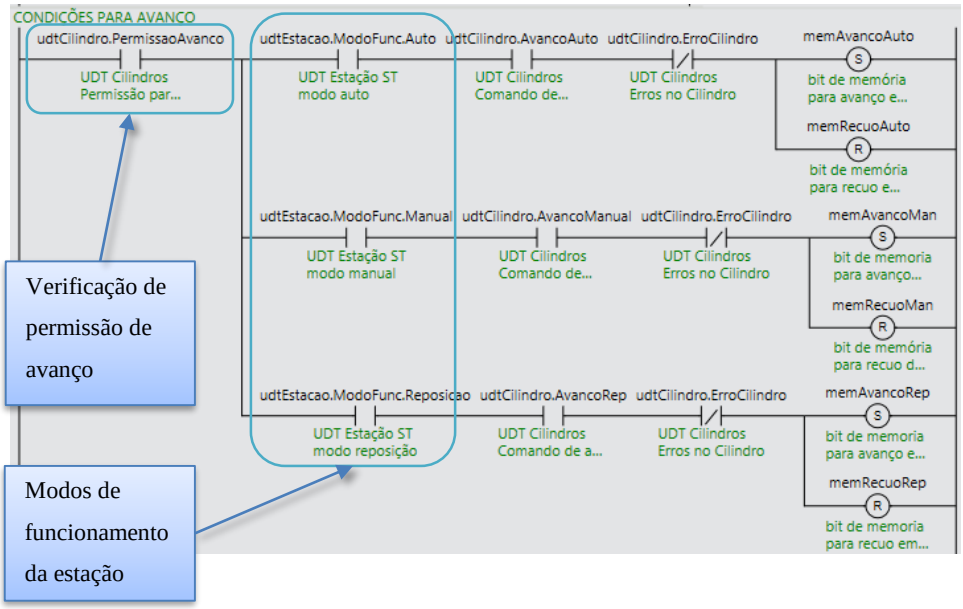


Figura 31 – Exemplo de seleção dos modos de operação para os diferentes modos de funcionamento da máquina/estação.

Dentro deste algoritmo também se preveem os erros que podem ocorrer nos cilindros. Estes erros são comuns a qualquer um dos modos de funcionamento e serão despoletados numa janela especifica da HMI. Previu-se que o cilindro pudesse ter quatro situações de inconformidade de funcionamento, das quais, caso se verifiquem o cilindro não realizará nenhuma operação. Passa-se a enumerar e descrever os quatro possíveis erros:

- "*TimeOut* Avanço";
- "*TimeOut* Recuo";
- "Sensores em Simultâneo";
- "Falta de permissão".

Os erros de "*TimeOut* Avanço" e "*TimeOut* Recuo" surgem caso as entradas associadas aos sensores de fim de curso de avanço e recuo do cilindro não sejam verdadeiras até um determinado tempo após a comando de avanço e recuo, respetivamente. No caso da ocorrência de um destes "*TimeOut*", o erro despoletado fica associado à variável respetiva ao sentido do movimento, (ver Figura 32), isto é, se o cilindro deveria avançar e esse tempo foi ultrapassado, então será despoletado o erro de "TimeOut Avanço". Este erro alerta para uma possível avaria do sensor ou electroválvula, podendo tratar-se também de um defeito ou obstrução no cilindro, ou qualquer outro motivo que impossibilite a devida translação do cilindro até à posição de fim de curso, definida pelo sensor.

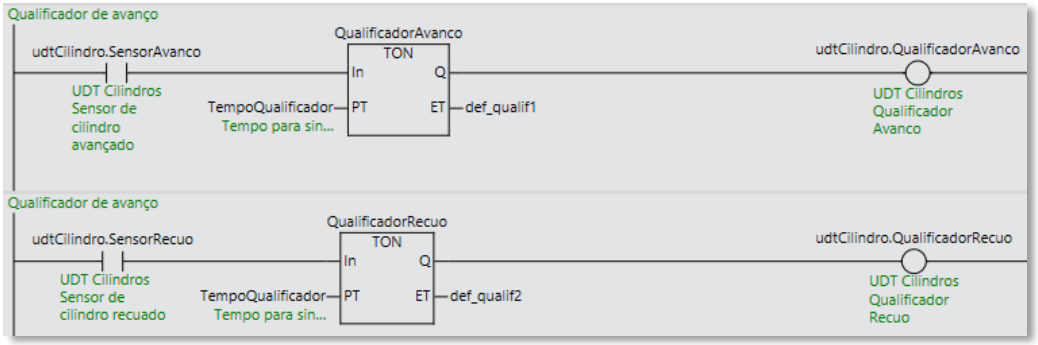


Figura 32 – Implementação do timer para deteção dos erros de "*TimeOut*" (linguagem *Ladder*).

O erro "Sensores em Simultâneo" é despoletado quando as duas entradas associadas aos sensores de fim de curso, estiverem ativas em simultâneo. Este erro prevê a avaria ou a fixação incorreta dos sensores que force a ativação de ambos simultaneamente.

A "Falta de Permissão" é originada quando é dado um comando de avanço ou recuo do cilindro sem que este tenha permissão para tal. As permissões são definidas nos programas das estações pelo programador, ver Figura 33.

Figura 33 - Exemplo de atribuição da "permissão de recuo" do cilindro 1.

O algoritmo de controlo dos cilindros pneumáticos foi desenvolvido numa *Function Block*. Optou-se pela utilização de uma *Function Block*, pois como já foi referido anteriormente, estes blocos permitem guardar o estado das variáveis internas durante a execução dos ciclos de trabalho do

autómato. A monitorização dos sensores do cilindro é feita através de uma *Function* inserida na *Function Block* principal.

Aplicação do módulo:

As *Function Block* de controlo dos cilindros pneumáticos (ver Figura 34), vêm facilitar o desenvolvimento de sistemas em que o número de atuadores deste tipo é elevado e, em termos de comparação, se a programação fosse feita pela técnica de "*copy & paste*" o tempo de desenvolvimento do programa geral seria previsivelmente superior e mais suscetível de surgir erros.

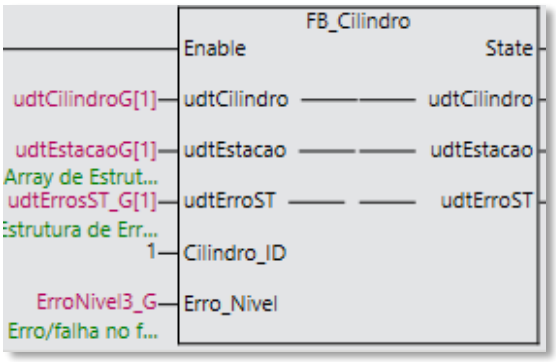


Figura 34 - *Function Block* de controlo do cilindro pneumático.

O acesso simplificado aos membros da estrutura de dados dos cilindros pneumáticos, permite que a elaboração de um programa seja facilitada. A estrutura de dados quando adicionada nas variáveis globais do programa deve ser declarada como sendo um *array* cujo número de elementos é igual ao número de cilindros. Desta forma, aceder aos dados de um cilindro é uma tarefa simples em que apenas será necessário alterar o índice do *array*, para identificar o cilindro. Ver exemplo da Figura 35.

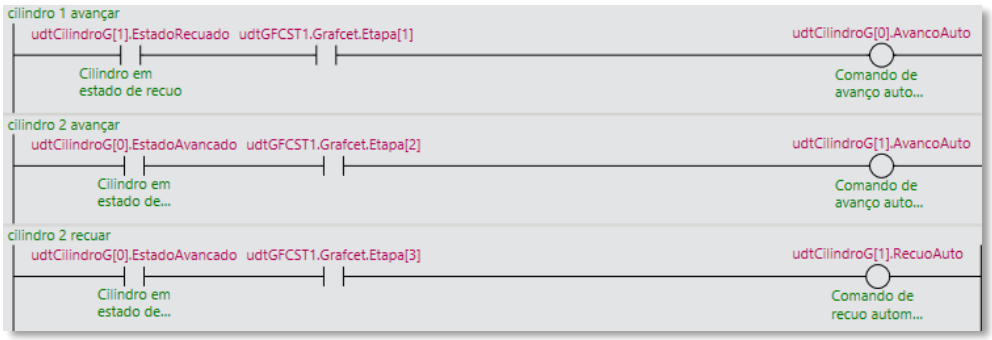


Figura 35 - Acesso à estrutura de dados do cilindro pneumático (*Sysmac Studio*).

Interface de comunicação com o utilizador:

Torna-se fundamental existir uma janela para controlar e monitorizar os cilindros pneumáticos. Sendo assim, a elaboração desta janela permite ao utilizador aceder a alguns dados do cilindro, como por exemplo: a posição, o estado, as permissões, os tempos de avanço e recuo, os erros, etc. Esta janela permite ainda realizar as operações de avanço e recuo do cilindro caso a estação onde se inserir o cilindro esteja em modo de funcionamento Manual. A Figura 36 apresenta um

exemplo de aplicação da janela padrão desenvolvida para monitorizar e controlar cilindros pneumáticos.

Figura 36 - Ecrã de "Monitorização e controlo do cilindro pneumático".

3.3.3. Controlo do prato rotativo

Os pratos rotativos (Figura 37) são estruturas compostas por ninhos e postos de trabalho. Os ninhos estão dispostos na periferia do prato rotativo e, têm a função de suportar as peças durante o processo. Os postos de trabalho estão localizados em redor do prato rotativo e funcionam como estações de trabalho.

O algoritmo desenvolvido para este módulo, possibilita a comunicação de pratos rotativos com controlador integrado. Estes controladores são desenvolvidos pelo fabricante dos pratos rotativos, e fazem o comando e controlo do motor que realiza a rotação. Para o caso especifico, a comunicação é feita por I/O, a rotação funciona por indexação, em que existe apenas um comando para iniciar a rotação, e cada vez que esse comando é dado, o prato roda uma posição. No fim da rotação o controlador do prato rotativo, comunica com o PLC indicando que se encontra em “fim de rotação”. Se ocorrer algum erro na rotação, o controlador comunica através de *Outputs* dedicados. Deste modo é possível monitorizar e identificar erros específicos detetados pelo controlador.

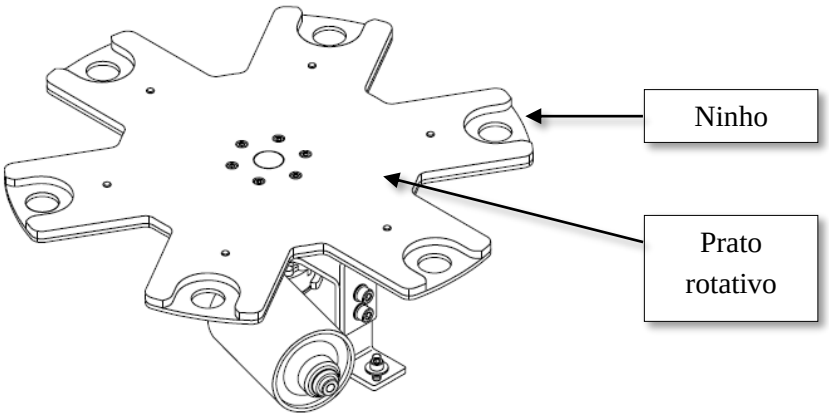


Figura 37 - Prato rotativo de 6 posições (*Festo Didactic indexing plate*) [27].

Programa geral:

Em termos do programa geral, tal como o cilindro pneumático, o prato rotativo tem diferentes modos de operação para os diferentes modos de funcionamento da máquina. Também, em semelhança ao cilindro pneumático, para cada um dos modos de funcionamento existe uma permissão de rotação e mais uma lista de condições que se devem verificar antes de iniciar a rotação. Considerando que esta estrutura (Prato rotativo) é semelhante ao de uma linha de montagem, em que apenas muda a configuração, pode-se considerar que a estrutura de dados para o controlo e monitorização dos postos de trabalho, funciona de forma semelhante à das estações de trabalho. Sendo assim, o controlo do modo de funcionamento dos postos é feito com recurso à *Function Block* de controlo das estações de trabalho. Isto é, na elaboração de um sistema que inclua um prato rotativo, o controlo dos postos de trabalho é feito da mesma forma que as estações de trabalho numa linha de montagem, o que implica que para cada posto de trabalho, seja necessário inserir uma *Function Block* de controlo de estações. Porém, foi necessário implementar uma *Function Block* para realizar o controlo do funcionamento dos postos de trabalho. Desta forma é possível o controlo individual dos postos de trabalho, o que implica também que, para cada posto, seja necessário incluir um módulo deste tipo.

O algoritmo de controlo do prato rotativo é a estrutura mais complexa do projeto, como tal, é necessário dividir a estrutura em módulos mais simples, decompondo o programa em operações específicas. A *Function Block* principal encapsula o algoritmo de controlo geral e, de forma hierárquica chama as *Function Blocks* e *Functions* responsáveis pelas operações específicas de controlo e monitorização do prato.

No conjunto, este algoritmo deve ser capaz de realizar as seguintes tarefas:

- Definir o número de postos no prato rotativo: 2 a 16;
- Os dados do processamento da peça devem acompanhar a mesma durante a rotação do prato;
- Opção de ativar/desativar os ninhos do prato;
- Implementação do sincronismo entre o prato e os postos;
- Rejeição de peças pelos postos;

- Verificação das peças no ninho após a abertura de uma porta e após colocação do prato rotativo em modo automático;
- Contabilizar o tempo de ciclo do prato e dos postos;
- Contabilizar o tempo de espera dos postos mais rápidos relativamente ao posto mais lento.

Para garantir a segurança do prato rotativo e do sistema envolvente, o algoritmo deve ainda prever os erros específicos, para garantir o bom funcionamento. O programa deve despoletar um erro e parar o funcionamento do prato rotativo, quando ocorre alguma irregularidade na rotação de prato ou nas peças. De seguida descreve-se alguns dos erros mais relevantes.

O primeiro erro é detetado pelo controlador do prato rotativo e assinala duas situações. A primeira situação ocorre quando o prato passa da posição de *Stop*, este erro pode dever-se a falha mecânica (desgaste dos travões) ou então o tempo de paragem é muito alto, este valor pode ser reajustável. A segunda situação é se o prato chega com um atraso de aproximadamente 10 segundos à posição de *Stop*, este caso resulta da situação de *Overload* do prato.

O segundo erro ocorre quando há um desaparecimento de uma peça de um dos ninhos. Este caso pode ocorrer, durante o processo por exemplo, em que por algum motivo a peça sai do respetivo ninho, ou então na abertura de uma porta se o operário retirar alguma peça do prato. O **Anexo 2** e **Anexo 3** descrevem os processos de rearme da porta e de verificação de erro por falta de peça.

Descrição dos processos:

Começa-se por apresentar o funcionamento do processo principal de controlo e gestão do funcionamento global do prato rotativo. Esse processo está representado através do GRAFCET que descreve o programa (ver Figura 38). Este GRAFCET foi implementado com duas macro etapa (M1 e M2). A macro etapa M1 representa a função que gere o funcionamento dos postos de trabalho. A macro etapa M2 representa a função que controla e gere a rotação do prato.

Processo principal de controlo do prato rotativo

O GRAFCET da Figura 38, descreve o funcionamento do prato rotativo no modo de funcionamento automático. Neste GRAFCET o programa está implementado da seguinte forma:

Etapa 0: o programa principal fica em espera até que o prato rotativo seja colocado em modo de produção. Após a colocação da estação do prato rotativo em modo de produção, é feita uma verificação dos ninhos. Essa verificação é feita como uma medida de prevenção para evitar que haja ninhos previamente ocupados, os processos de iniciação e verificação do prato rotativo estão descritos nos GRAFCET dos **Anexo 4** e **Anexo 5**. De seguida é feita a verificação de existência de uma nova peça "pronta para ser processada" pelo primeiro posto de trabalho.

Etapa 1: após a verificação da primeira recetividade, o GRAFCET avança para a primeira macro etapa. Nesta fase, o processo principal dá o comando de início do ciclo de trabalho aos postos de trabalho. Com o início de ciclo de trabalho dos postos, o processo principal fica em "espera" e permanece na etapa 2.

Etapa 3: dá-se a passagem para a segunda macro etapa (M2) quando se verificar o fim de ciclo de todos os postos de trabalho. Esta verificação só é verdadeira quando todos os postos que

constituem o prato rotativo, tenham terminado os seus ciclos de trabalho. Nesta macro etapa, inicia-se a rotação do prato rotativo e as peças são levadas para os postos seguintes.

Etapa 4: no período de execução da função de controlo de rotação do prato, o processo principal fica em "espera". O GRAFCET finaliza e retoma a etapa 0, no momento em que termina a rotação do prato.

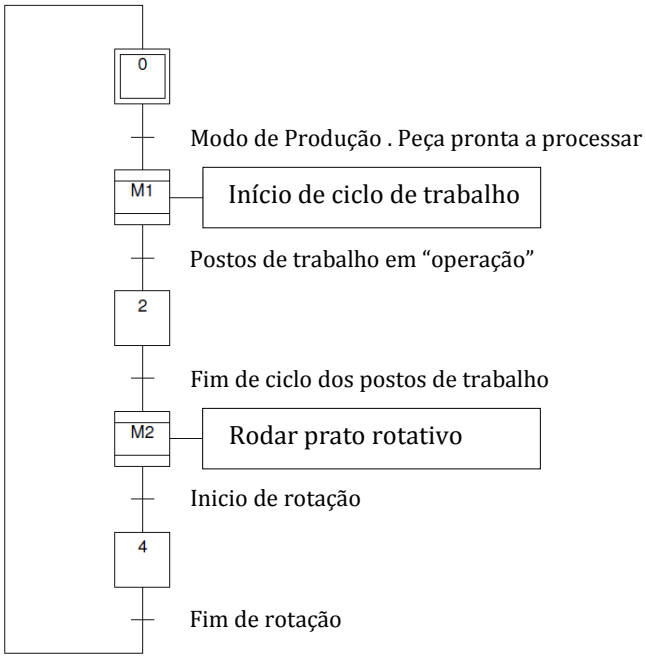


Figura 38 - GRAFCET de controlo e gestão do prato rotativo em modo automático.

Processo de controlo das operações do posto de trabalho

O funcionamento da *Function Block* de controlo do posto de trabalho é representado através do GRAFCET da Figura 39. Este GRAFCET descreve o programa de controlo individual de cada posto de trabalho do prato rotativo. O processo de controlo do funcionamento dos postos de trabalho segue as seguintes etapas:

Etapa 0: este programa inicia quando recebe o sinal de "início de ciclo de trabalho". Este sinal é proveniente do processo principal de controlo do prato rotativo.

Etapa 1: esta etapa de passagem, verifica a condição da peça. Caso a peça tenha condições de ser processada, o GRAFCET evolui para a etapa 2, caso contrário, o GRAFCET evolui para a etapa 4.

Etapa 2: a passagem para esta etapa, ocorre quando o posto tiver uma peça "pronta a processar". Nesse caso, dá-se o comando de início de operação.

Etapa 3: a passagem para esta etapa, ocorre com o fim das operações do posto. Nesta situação, o programa de controlo do posto de trabalho comunica com o programa principal, a indicar que o processo a desempenhar na peça terminou.

Nesta fase o programa de controlo do posto fica à espera do momento em que se inicia a rotação do prato rotativo. Nesse momento, o GRAFCET retoma o estado inicial e fica à espera de um novo comando de "início de ciclo de trabalho".

Etapa 4: esta etapa é ativa quando existe alguma imposição que impeça o início das operações designadas ao posto de trabalho. Este impedimento de operação pode surgir no caso de não existir peça, ou caso exista uma peça danificada, entre outros motivos.

Neste caso, se a peça não tiver condições para realizar as operações do posto, o programa fica à espera que terminem os processos dos restantes postos.

Etapa 5: nesta etapa o programa de controlo do posto, fica à espera do momento em que se inicia a rotação do prato rotativo, para que o GRAFCET retome o estado inicial.

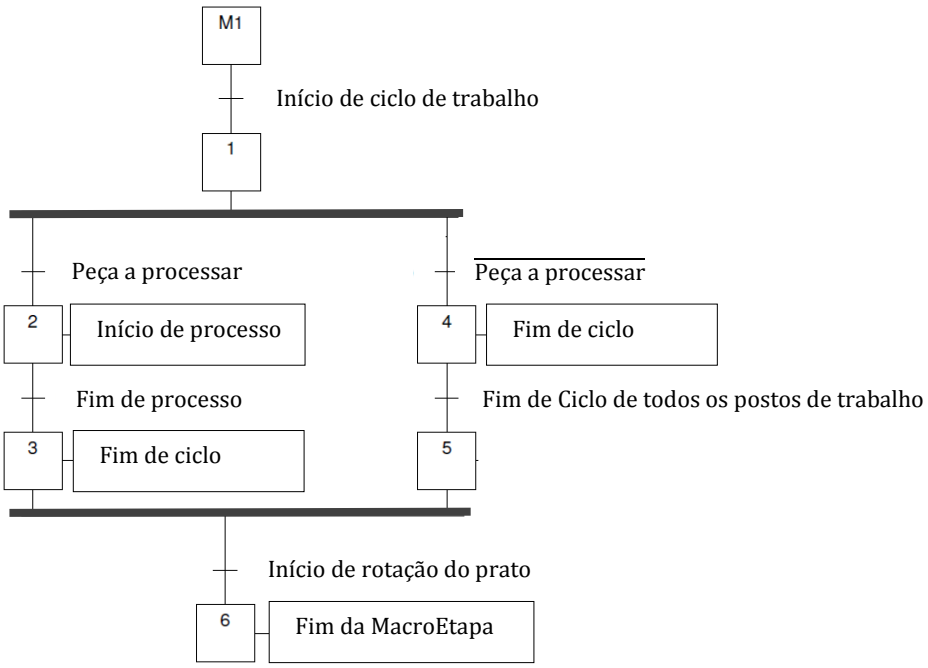


Figura 39 - GRAFCET de controlo e gestão do posto de trabalho.

Processo de controlo da rotação do prato

O GRAFCET da Figura 40 representa o processo de controlo da rotação do prato rotativo. A *Function Block* de controlo deste processo é chamada no programa principal de controlo do prato rotativo, com a ativação da macro etapa M2. O GRAFCET de controlo e monitorização da rotação do prato rotativo segue as seguintes etapas:

Etapa 0: o programa inicia com o comando de "Início de rotação" do programa principal de controlo do prato rotativo.

Etapa 1: nesta etapa o programa ativa a variável de comando de início de rotação. Esta variável comunica ao controlador do prato rotativo que este deve iniciar a rotação.

Etapa 2: uma vez iniciada a rotação do prato, esta etapa fica ativa quando o sensor de posição do prato deixa de ter o sinal ativo. Nesta etapa o programa fica em "espera", enquanto o prato se encontre em rotação.

Etapa 3: a ativação desta etapa verifica-se quando recebe o sinal do sensor de posição. Este sinal indica que o prato indexou uma posição. Neste momento o programa envia para o controlador do prato rotativo o comando de paragem. De seguida comunica ao programa principal que a rotação terminou. Após a verificação de fim de rotação, o programa retoma o estado inicial

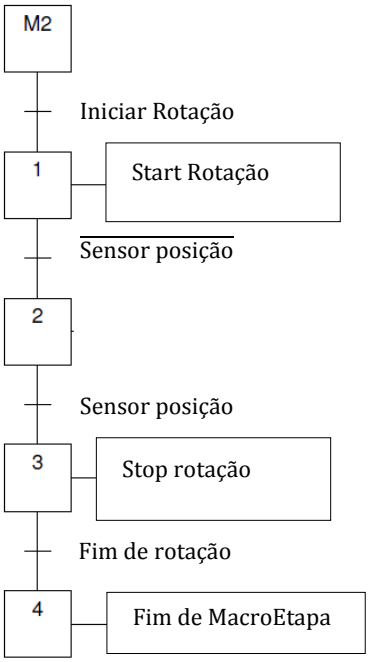


Figura 40 - GRAFCET de controlo e gestão da rotação do prato.

Aplicação do módulo:

No momento da inserção da *Function Block* de controlo do Prato Rotativo num programa, o programador deve inserir os parâmetros no bloco que definam o número de ninhos e postos. Pois um prato pode ser constituído por diferentes números de postos e ninhos, dependendo da aplicação, sendo que o máximo definido pela empresa Siroco tenha sido de 16 ninhos e 16 postos de trabalho.

O algoritmo foi então implementado de maneira a permitir que a gestão dos dados seja dinâmica, isto é, no tratamento dos dados dos ninhos e dos postos (Ocupação dos ninhos, Estado de funcionamento, etc.), recorreu-se à utilização de *arrays* para guardar as informações necessárias. Para tal, os *arrays* criados são de 16 posições (máximo definido) e os valores inseridos pelo programador para definir o número de ninhos e postos definem até que posição do *array* é feita a verificação do que se pretende analisar.

A *Function Block* de controlo do prato rotativo, tem a capacidade de executar e monitorizar todas as atividades de um prato rotativo do tipo que a empresa Siroco utiliza. Num só bloco, tornou-se mais simples a inserção do algoritmo de controlo de um prato rotativo na elaboração do programa geral de uma máquina. Com apenas a utilização da *Function Block* de controlo do prato rotativo

passa a ser possível controlar a rotação fazer a gestão dos postos e ninhos, monitorizar os erros, entre outras funcionalidades. O bloco do prato rotativo funciona como uma estação de trabalho, que pode ser independente ou inserido numa linha de montagem. Para tal o programa geral do bloco precisa de aceder às estruturas de dados da estação, do prato rotativo e dos erros, ver Figura 41.

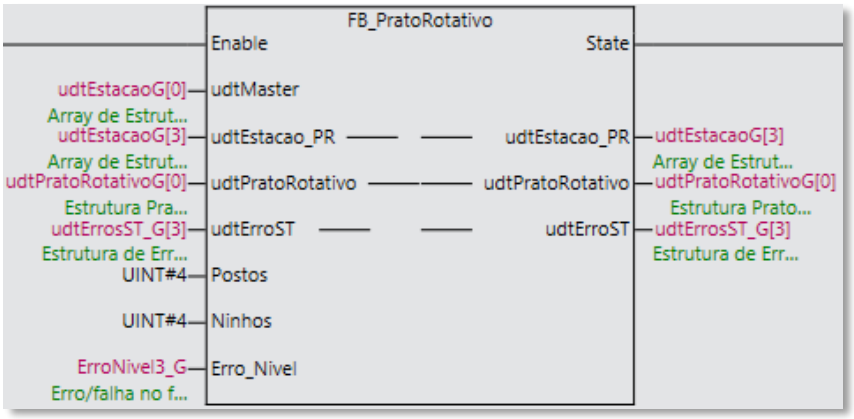


Figura 41 - *Function Block* de controlo do prato rotativo.

A sincronização do funcionamento dos postos de trabalho é feita na *Function Block* de controlo do prato rotativo. Durante o modo de funcionamento automático (modo de produção), existe a interação entre os postos de trabalho e o prato rotativo. A comunicação entre os dois módulos de controlo é feita atraves da estrutura de dados do prato rotativo. Na Figura 42 apresenta-se um exemplo do aspecto exterior da *Function Block* de controlo de um posto de trabalho.

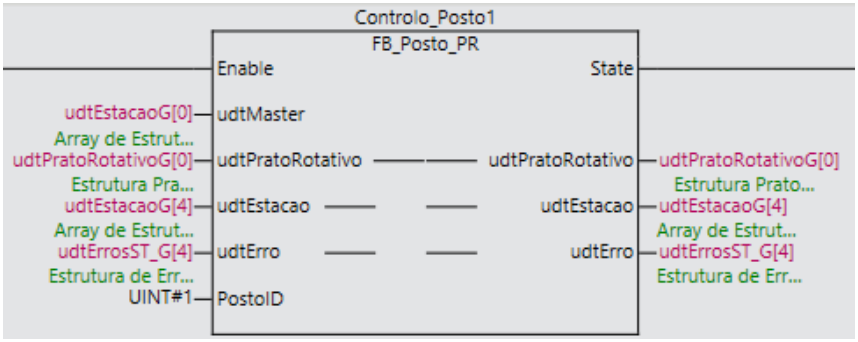


Figura 42 - *Function Block* de controlo dos postos de trabalho.

Interface de comunicação com o utilizador:

Para monitorização de alguns dados, foram implementadas duas janelas na HMI para o prato rotativo, que permitem ao utilizador verificar informações do estado dos postos de trabalho, ninhos e adquirir dados estatísticos da produção (tempos de ciclo, dados de processamento, etc.). Existe ainda uma terceira janela que permite selecionar o modo de funcionamento e comandos específicos do prato rotativo, tal como é feito para as estações de trabalho de uma linha de montagem. Para cada posto de trabalho, o procedimento é semelhante às estações de trabalho apresentadas anteriormente. Cada posto possui uma janela individual para monitorizar dados estatísticos e controlar os modos de funcionamento do posto.

3.4. Gestão de alarmes

Um alarme é uma notificação ao operador sobre a ocorrência de uma anomalia ou erro que necessita de intervenção para retornar ao funcionamento normal do sistema. Um sistema de alarmes tem a função de detetar e notificar ao operador o surgimento de um erro ou anomalia. No entanto, é necessário existir uma correta identificação de anomalia e, para tal, é preciso uma correta interpretação de múltiplos alarmes.

Tal como já foi referido anteriormente, a deteção de erros é feita dentro de cada um dos blocos funcionais desenvolvidos. Desta forma, a verificação de existência de erros é individual, podendo atuar diretamente no próprio programa. Caso se verifique a ocorrência de algum erro ou anomalia, os próprios blocos realizam uma determinada ação, como por exemplo, a sinalização de um alarme e/ou a paragem local do sistema.

Porém, a ineficiência dos sistemas de alarme acontece, geralmente, por falta de uma metodologia de análise e manutenção das configurações dos alarmes, que como tal é algo a evitar. Para melhorar a eficiência na resolução dos alarmes, foi proposto pela empresa siroco, a hierarquização dos alarmes por níveis. O método proposto baseia-se na filosofia de que todo o alarme deve ser munido de um grau de prioridade que facilite o diagnóstico do alarme. Seguindo esta filosofia, com o surgimento de um erro/anomalia a máquina deve estar em modo "pausa", após a identificação do alarme, este deve ser interpretado e despoletado, caso seja o alarme com maior grau de prioridade. Deve ainda apresentar as informações mais importantes com um conjunto de respostas que permitam uma tomada de decisão correta por parte do operador.

Como tal, é necessário desenvolver uma estrutura de dados para os alarmes. Esta estrutura é tida como uma variável de passagem que é comum a todos os blocos funcionais principais que sejam introduzidos no projeto (Ver Figura 43). É nesta estrutura que será registada qualquer ocorrência que surja e qual o seu nível de prioridade. Contudo, a atribuição da prioridade deve ser feita aquando da parametrização do bloco funcional. Essa classificação deverá ser feita em função da classificação dos níveis estabelecidos na empresa, ver exemplo da Figura 43.

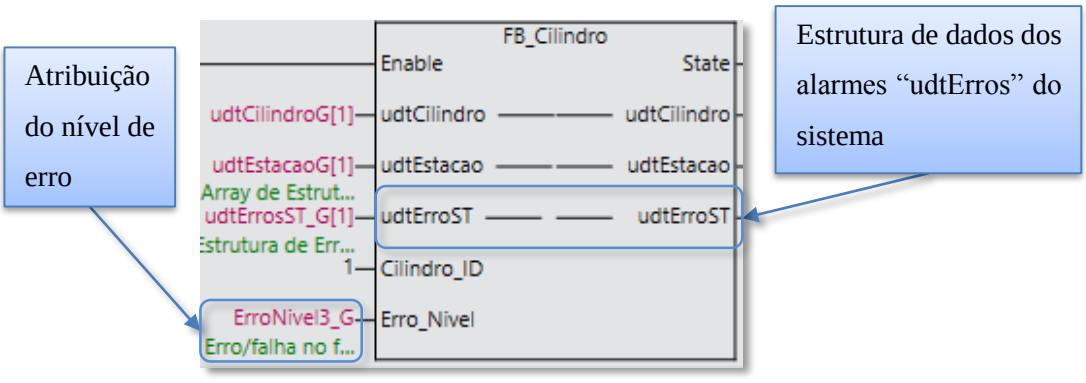


Figura 43 - Exemplo de atribuição do nível de erros do cilindro pneumático.

Deve ainda existir uma estrutura de dados dos alarmes associada a cada estação de trabalho. Para tal, ao adicionar a estrutura de dados dos alarmes nas variáveis globais do programa, deve ser declarada como sendo um *array* cujo número de elementos é igual ao número de estações do

equipamento. Desta forma, cada estação faz a gestão dos alarmes que ocorram nos periférico ou na própria estação.

Um sistema de alarmes tem a função de detetar e notificar a ocorrência de um erro ou anomalia. No entanto, é necessário existir uma correta identificação e descrição dos eventuais erros. A deteção de um erro é feita no interior de cada FB ficando registada num array que está associado à estação de trabalho. No entanto, existe a possibilidade de surgir mais do que um alarme com a mesma anomalia. O que dificulta a interpretação dos alarmes. Para resolver este tipo de situação, forma definidos diferentes níveis de erro. Esta hierarquia serve para atribuir a cada tipo de erro um grau de prioridade. Deste modo, se surgirem vários alarmes do mesmo tipo em simultâneo, apenas será despoletado o de maior grau.

A estrutura de dados para os alarmes, é composta por dois *arrays* com 150 posições cada. O primeiro para registar a ocorrência e o segundo regista o nível de prioridade do alarme. O nível de prioridade serve para A cada posição dos *arrays* corresponde um tipo de alarme. As primeiras 100 posições do *array* correspondem a alarmes dos cilindros pneumáticos, as restantes correspondem a alarmes da estação de trabalho. Ver na Figura 44, o exemplo de um registo de um alarme, ocorrido num determinado cilindro de uma estação.

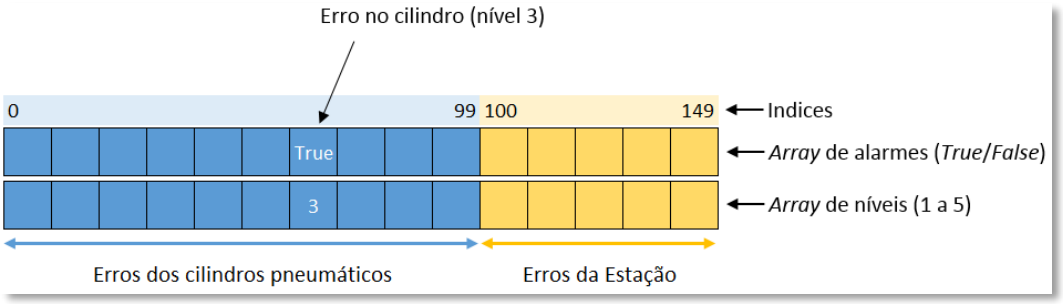


Figura 44 – Exemplo de registo de um erro num cilindro pneumático.

Seguindo ainda o exemplo do cilindro pneumático. A Figura 45 apresenta a parte do algoritmo de deteção de erros, que regista a ocorrência do erro do tipo "*TimeOut* Avanço". Quando surge um erro deste tipo, o programa regista a ocorrência e o nível de prioridade nos *arrays* de alarmes e níveis respetivamente, caso contrário o valor registado é nulo. Estes dados ficam registados nos *arrays* da estrutura de dados de alarmes da estação onde estiver inserido o cilindro pneumático. A alocação dos erros no *array* depende do resultado da expressão de cálculo (1). Em que "n_total_erros" é o número total de erros de um cilindro pneumático, o "Cilindro_ID" é o número de identificação do cilindro e, o "Erro_ID" é o número de identificação do erro.

$$j = n_{total_{erros}} \times Cilindro\_ID + Erro\_ID - 1 \quad (1)$$

Por exemplo, pressupondo que:

- o nº total de erros de um cilindro pneumático é "4";
- o número de identificação do cilindro é "0";

- o número de identificação do erro é 1.

Substituindo esses valores na expressão de cálculo (1), o resulta é "0", portanto o erro ficará alocado no índice "0" do *array*.

```
//ERRO TIMEOUT AVANÇO:
IF(udtCilindro.TimeOutAvanco) THEN
    j:=n_total_erros*(Cilindro_ID)-1+1; //erro_index=0
    udtErroST.Ar_Erro[j]:=TRUE;
    udtErroST.Ar_Nivel[j]:=Erro_Nivel;
ELSE
    j:=n_total_erros*(Cilindro_ID)-1+1;
    udtErroST.Ar_Erro[j]:=FALSE;
    udtErroST.Ar_Nivel[j]:=Erro_Nivel;
END_IF;
```

Figura 45 – Caso particular do algoritmo para deteção e registo de erros (Linguagem Structured Text).

Na Figura 46 é apresentado o algoritmo implementado para a deteção de alarmes. O processo de verificação de ocorrência de alarmes é feito na *Function Block* de controlo das estações de trabalho. Este algoritmo verifica o *array* que regista os alarmes, isto é, com um ciclo *for* percorre todas as posições do *array* e, caso detecte alguma ocorrência interrompe a verificação e regista o evento, caso contrário termina o ciclo. Esta verificação é feita ciclicamente e, caso detecte alguma ocorrência coloca a estação em "Pausa".

```
Len_Arry:=SizeOfAry(udtErro.Ar_Erro);   (*Dimensão do array de alarmes*)
Num_Erros:=0;                           (*Variável para "contar" o nº de erros*)
FOR Index:=0 TO Len_Arry-1 BY 1 DO      (*Verificação*)
    IF udtErro.Ar_Erro[Index]=1 THEN
        Num_Erros:=Num_Erros+1;
        EXIT;
    ELSE
        Num_Erros:=0;
    END_IF;
END_FOR;

IF Num_Erros>0 THEN                     (*Registo do Erro*)
    udtEstacao.Eventos.Erro:=TRUE;
ELSE
    udtEstacao.Eventos.Erro:=FALSE;
END_IF;
```

Figura 46 - Algoritmo para deteção de erros (Linguagem *Structured Text*).

Na ocorrência de novos alarmes, os erros registados na estrutura de dados dos alarmes do sistema são, posteriormente, despoletados numa janela de monitorização de alarmes da HMI (ver Figura 47). Nesta janela, a forma de visualização dos alarmes é feita em lista. É nesta janela que será feita a seleção de erros que posteriormente são apresentados na lista de alarmes. Esta seleção tem por base o grau de prioridade, ou seja, se houver uma falha na alimentação de ar comprimido da máquina, esta falha será registada e ser-lhe-á atribuído um grau de prioridade superior ao erro de "*TimeOut*" do movimento de um cilindro pneumático que se encontrasse em translação visto que este último erro trata-se de uma consequência da falha na alimentação de ar comprimido. A adoção deste método

trás, ao sistema de gestão de alarmes uma redução da sobrecarga de alarmes e vem permitir uma mais fácil interpretação da origem dos erros/anomalias. Na janela de monitorização de alarmes, existe ainda a possibilidade de navegar para a janela de "Informação dos Alarmes" (ver Figura 48). Para tal, basta selecionar o alarme e pressionar no botão "Ajuda" situado no lado esquerdo do ecrã. Nessa janela está implementado em "*background*" um algoritmo, implementado em linguagem "*Visual Basic*", que identifica e despoleta informações como:

- Identificação do alarme;
- Estação onde ocorreu o erro;
- O número de identificação do erro;
- Descrição detalhada do erro.

Existe ainda a possibilidade de adicionar ficheiros em formato PDF e/ou vídeos, de como resolver o problema.

Figura 47 - Ecrã de "Janela de Alarmes".

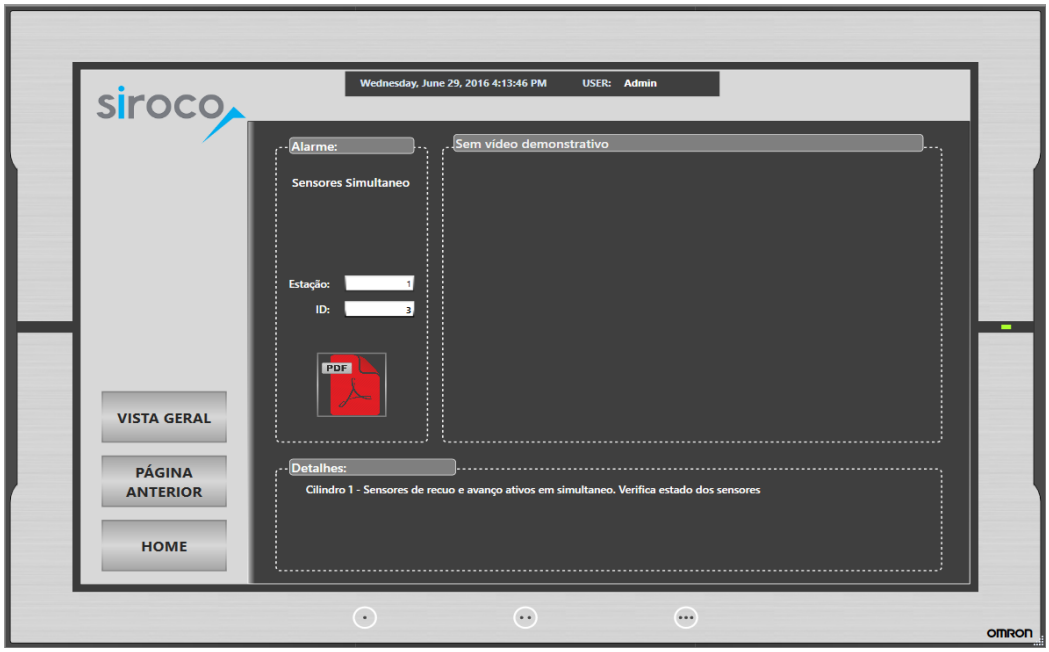


Figura 48 - Ecrã de "Informação dos Alarmes".

Resumindo, a gestão da atribuição de níveis de prioridade é feita no PLC, na fase de parametrização dos blocos funcionais e a seleção e apresentação dos alarmes é feita na HMI.

4. CASOS DE ESTUDO

Neste capítulo são apresentados dois casos de estudo com particularidades distintas. O primeiro caso de estudo trata-se de uma linha para enchimento, colocação de tampa e rotulagem de garrafas (ver *layout* na Figura 49). Neste primeiro caso procura-se idealizar um equipamento em que seja possível incluir todos os módulos de controlo num só programa. O segundo caso de estudo trata-se de um equipamento real implementado pela empresa Siroco, que tem como função inserir terminais de HV e díodos em *secondary spools* (ver *layout* na Figura 80). O facto de no segundo caso de estudo se ter utilizado um equipamento real, permite ir de encontro ao objetivo principal do projeto, que consiste em desenvolver módulos de controlo de estruturas específicas, que possam ser utilizados no desenvolvimento dos programas de controlo dos equipamentos produzidos pela empresa.

O objetivo de realizar testes aos blocos funcionais desenvolvidos em dois casos de estudo diferentes centra-se, essencialmente, na demonstração e na consecutiva validação do uso de blocos funcionais na programação de equipamentos de montagem de componentes, no que diz respeito à sua versatilidade (capacidade de adaptação), eficiência e capacidade de reutilização. A implementação destes casos de estudo permite reforçar a ideia de que é possível desenvolver algoritmos de controlo para equipamentos com diferentes caraterísticas, seguindo as metodologias propostas para o desenvolvimento de programas de controlo para equipamentos de montagem de componentes. Contudo, dado o difícil acesso a sistemas reais que sejam facilmente reconfigurados e sem todos os custos e riscos associados, os ensaios dos cenários serão realizados em modo simulado. Deste modo será igualmente possível demonstrar e validar o uso das janelas/objetos padrão da interface e dos módulos de controlo desenvolvidos.

Para a implementação dos casos de estudo, a ferramenta de desenvolvimento é o *Sysmac Studio*. Este *software* permite ao utilizador simular um conjunto de dispositivos, como por exemplo: PLC e HMI. Utilizando o simulador do *Sysmac Studio* é possível obter um ambiente de execução do programa, integrando o controlador e os ecrãs de interface de controlo, facilitando então a análise do sistema antes da montagem do sistema real e reduz o tempo necessário para o desenvolvimento e arranque do equipamento.

4.1. Caso de Estudo 1 – Linha de Enchimento e Rotulagem de Garrafas

4.1.1. Descrição geral

No primeiro caso de estudo, apresenta-se um projeto para uma aplicação de uma linha para o enchimento de garrafas. Este equipamento é composto por oito atuadores pneumáticos, um autómato programável, uma consola HMI, uma estação de abastecimento (estação 0 - Figura 49), uma estação de sopro (estação 1), uma estação de enchimento (estação 2), uma estação de colocação de tampa na garrafa (estação 3) e, por fim, uma estação de colocação e verificação de rótulo (estação 4). A última estação de trabalho trata-se de um prato rotativo, com quatro postos de trabalho. O primeiro posto é o ponto de abastecimento do prato rotativo, o segundo é um posto de rotulagem das

garrafas, o terceiro é um posto de verificação da colocação do rótulo e por último o posto de descarga das garrafas. O prato possuí quatro ninhos que têm como objetivo transportar as garrafas pelos quatro postos dispostos em redor do prato rotativo.

O equipamento é semiautomático, o operador, caso necessário, poderá intervir de acordo com algum alarme que ocorra.

Num primeiro momento, quando se inicializar o equipamento, as estações estarão livres e à espera de garrafas. Sendo assim, a primeira garrafa a entrar na linha terá acesso direto à primeira estação, uma vez que esta se encontrará em estado de "espera de garrafa" para iniciar o seu processo. Mas se a fluência de entrada de garrafas for elevada, estas poderão estar sujeitas a tempo de espera caso a cadência da linha não seja suficiente. Esse controlo de fluência das garrafas é ditado pelo sincronismo de funcionamentos dos *Stoppers* dispostos ao longo da linha. Estes *Stoppers* têm a função de controlar a circulação das garrafas pela linha. Cada estação possui dois *Stoppers*, um à entrada da estação e outro à saída. O *Stopper* de entrada e de saída estarão ativos caso a estação esteja ocupada com garrafa e, por conseguinte, permitir à estação executar as suas tarefas na garrafa que se encontra na sua área de trabalho, e para impedir a entrada de mais garrafas na sua área de operação. Assim que a estação termina a sua tarefa, o *Stopper* de saída é desativado permitindo assim a saída da garrafa. Depois da saída da garrafa com a operação concluída, o *Stopper* de saída volta a ser ativo e o *Stopper* de entrada é desativado permitindo a entrada de uma nova garrafa. O Processo de sincronização dos *Stoppers* repete-se em todas as estações, exceto no prato rotativo que apenas possui um *Stopper* na sua entrada.

O processo de funcionamento das estações inicia-se com a chegada da garrafa às suas áreas de operação. Quando a estação verifica que se encontra uma nova garrafa pronta para ser processada, verifica se a posição da garrafa na palete é a correta, antes de dar início ao processo. A operação de verificação de posição é feita com recurso a sensores que verificam se existe algum mau posicionamento que possa pôr em causa o bom desempenho do processo. Sendo assim, as estações de trabalho apenas operam caso as garrafas se encontrem bem posicionadas, caso contrário a garrafa fica inoperável.

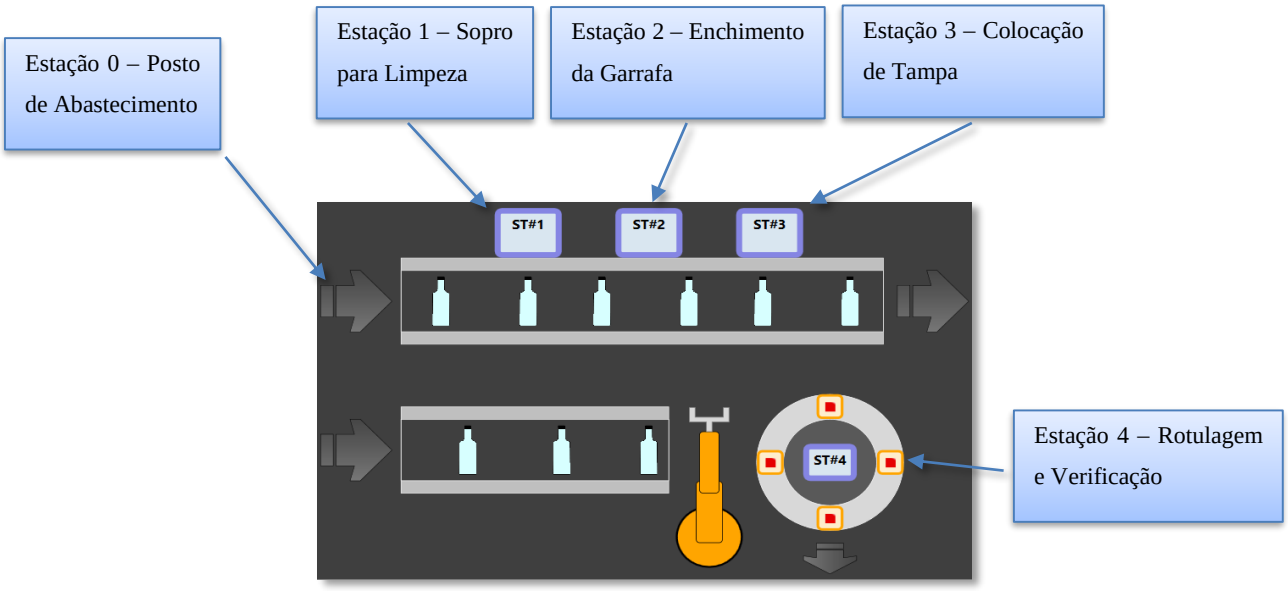


Figura 49 - *Layout* do Caso de Estudo 1.

A sequencia de funcionamento do equipamento consiste:

- Estação 0 – Posto de abastecimento de garrafas – Operação manual;
- Estação 1 – Sopro de limpeza da garrafa – Operação automática;
- Estação 2 – Enchimento da garrafa – Operação automática;
- Estação 3 – Colocação de tampa da garrafa – Operação automática;
- Estação 4 – Colocação e verificação do rótulo – Operação automática.

A sequencia de funcionamento das estações consiste:

- Estação 0:
  - ✓ Não aplicável;
- Estação 1:
  - ✓ Cilindro 1 – Avança – Aperto do fixador da garrafa;
  - ✓ Cilindro 2 – Avança – Inserção da cabeça de sopro na garrafa;
  - ✓ Cilindro 2 – Recua – Retira a cabeça de sopro;
  - ✓ Cilindro 1 – Recua – Solta o fixador da garrafa.
- Estação 2:
  - ✓ Cilindro 3 – Avança – Aperto do fixador da garrafa;
  - ✓ Cilindro 4 – Avança – Inserção da cabeça de enchimento na garrafa;
  - ✓ Cilindro 4 – Recua – Retira a cabeça de enchimento;
  - ✓ Cilindro 3 – Recua – Solta o fixador da garrafa.
- Estação 3:
  - ✓ Cilindro 5 – Avança – Aperto do fixador da garrafa;
  - ✓ Cilindro 6 – Avança – Inserção da tampa na garrafa;
  - ✓ Cilindro 6 – Recua;
  - ✓ Cilindro 5 – Recua – Solta o fixador da garrafa.
- Estação 4:
  - ✓ Cilindro 7 – Avança – Colagem do rótulo na garrafa;

- ✓ Cilindro 7 – Recua;
- ✓ Cilindro 8 – Rotação – Verificação da garrafa;
- ✓ Cilindro 8 – Rotação – Retoma posição inicial cilindro pneumático.

4.1.2. Criação dos POUs

A organização escolhida para a criação dos POU, define a divisão dos programas em diferentes grupos de controlo. O primeiro grupo define o controlo geral da máquina, ou seja, as secções definidas neste grupo incluem as *Function Blocks* de controlo das estações e do sincronismo da linha. Seguem-se os grupos de controlo das estações, que serão quatro grupos, um para cada estação. As secções destes grupos de controlo das estações incluem as *Function Blocks* de controlo dos cilindros pneumáticos, os programas de reposição, de controlo e os programas para verificação das permissões. Por fim, o grupo com as secções de controlo do prato rotativo; as secções deste último grupo incluem as *Function Blocks* de controlo do prato rotativo, controlo dos postos de trabalho e as *Functions* de registo dos dados de leitura e contagem dos tempos de ciclo dos postos de trabalho. A Figura 50 permite visualizar a árvore de navegação dos POUs do programa desenvolvido para o controlo do equipamento.

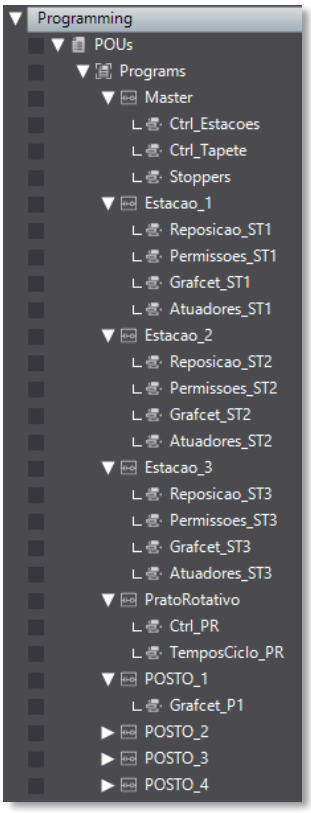


Figura 50 - Árvore de navegação dos POUs do caso de estudo 1.

4.1.3. Criação da interface

A interface Homem-Máquina foi desenvolvida de acordo com o modelo definido na secção 3.1. O desenvolvimento da interface deste caso de estudo, seguiu a estrutura de ecrãs, esquematizada

na Figura 51. Como se pode ver pelo esquema de ecrãs, a janela principal (1), dá acesso às janelas principais do programa, nomeadamente, a janela "Modo Funcionamento Geral", "Vista Geral", "Interface c/ Operador", "Estatísticas", "Configurações" e "Eventos".

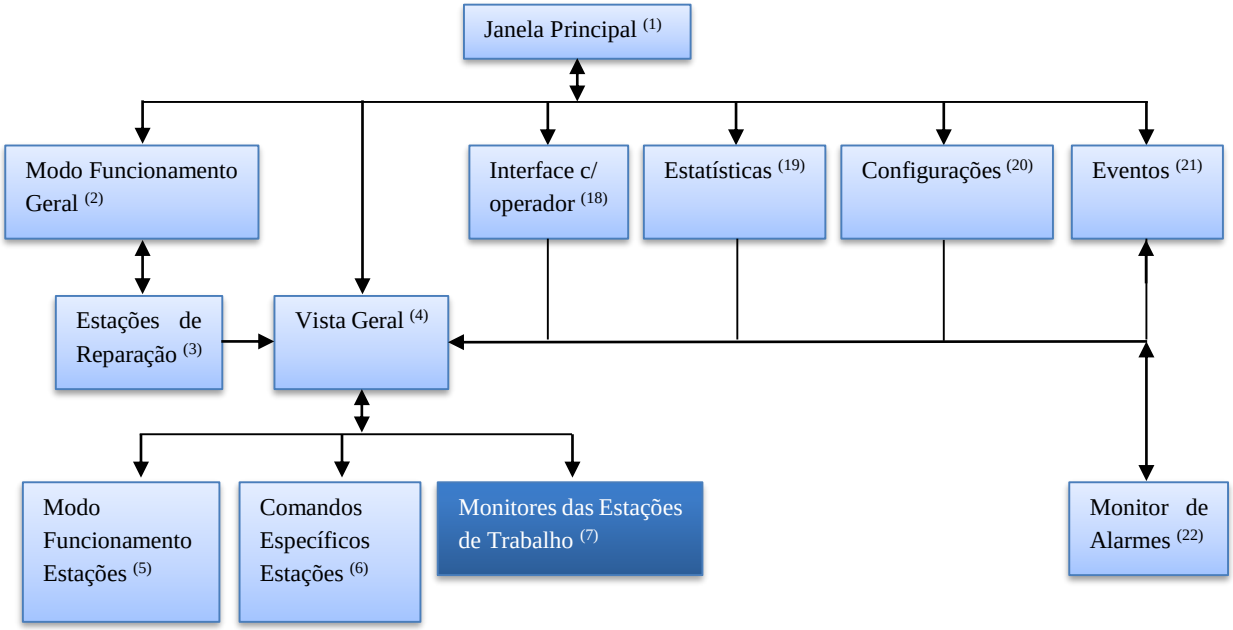


Figura 51 - Estrutura de ecrãs da consola do caso de estudo 1.

A janela de "Modo Funcionamento Geral", permite ao operador selecionar o modo de funcionamento do equipamento e, eventualmente, os comandos específicos da máquina. A comunicação entre o PLC e a janela de "Modo Funcionamento Geral" é feita através da estrutura de dados de controlo da estação, onde os botões de seleção dos modos e comandos estão associados a variáveis membros da estrutura de dados.

Os modos de funcionamento do equipamento, que podem ser o modo "Automático", "Manual" e "Reposição", permitem-nos ainda colocar a máquina em "Pausa", após a iniciação do modo de funcionamento "Automático". A janela "Vista Geral", permite ao operador aceder às janelas de controlo e monitorização das estações de trabalho. A janela "Interface c/ Operador", é a janela pela qual o operador se deverá "guiar", ou seja, em modo de funcionamento automático, o operador deverá acompanhar o processo através desta janela, pois esta janela fornece ao operador indicações dos procedimentos que deve seguir para acompanhar o funcionamento da máquina. A janela de "Eventos" permite ao operador visualizar e reconhecer alarmes que surjam durante o funcionamento do equipamento. A janela de eventos, permite ainda aceder a uma janela de "Ajuda", na qual serão apresentadas informações sobre o alarme e descrições e demonstrações (escritas e/ou vídeo) de como solucionar a anomalia. Qualquer janela da consola permite um acesso direto à janela "Vista Geral" e à janela "Eventos", para permitir uma mais fácil e rápida navegação pelos ecrãs da interface.

Na Figura 52, apresenta-se a estrutura de janelas de monitorização e controlo das estações de trabalho. São incluídas neste grupo de janelas as quatro estações referentes ao equipamento em causa (caso de estudo 1). As janelas (8), (10) e (12) são relativas aos ecrãs de monitorização e controlo das estações da linha de produção. A janela (14) apresenta o monitor do prato rotativo, que

permite o acesso às janelas de controlo e monitorização do prato rotativo, bem como às janelas dos postos de trabalho. A comunicação entre o PLC e a janela de monitorização e controlo do prato rotativo é feita através da estrutura de dados de controlo do prato rotativo.

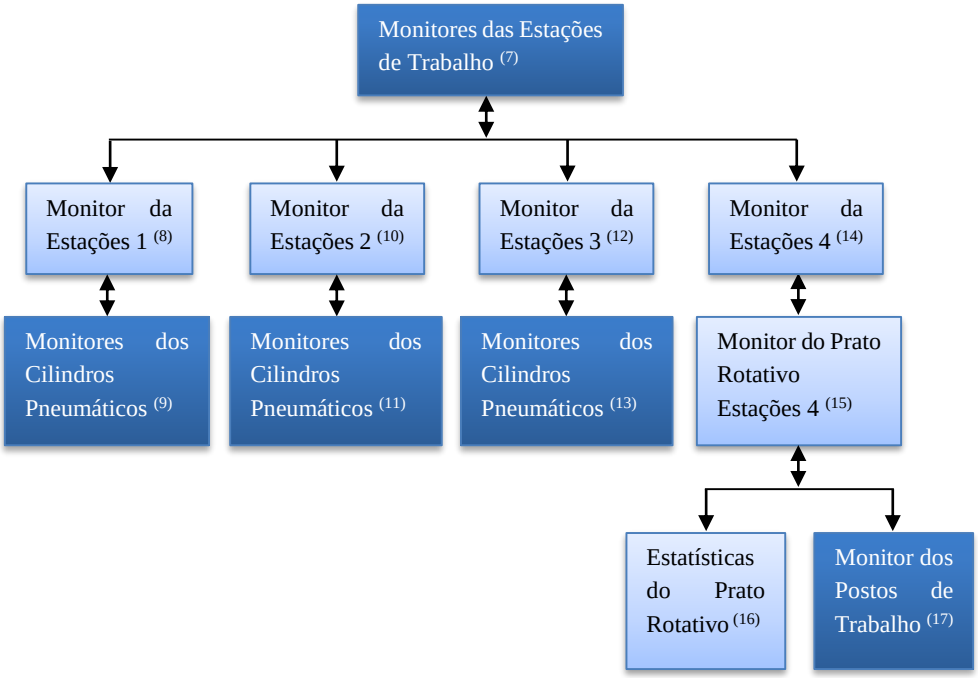


Figura 52 - Estrutura de ecrãs do grupo "Monitor das Estações de Trabalho".

Em algumas das janelas apresentadas anteriormente, são utilizados objetos do tipo IAG. Um exemplo é a IAG de monitorização dos cilindros pneumáticos que é incluída em todas as janelas de monitorização dos cilindros pneumáticos, mas também nas janelas de monitorização das estações de trabalho. Contudo, foram também incluídas IAGs do tipo "janela padrão", nomeadamente para as janelas de controlo e monitorização das estações de trabalho, do prato rotativo e dos cilindros pneumáticos. As IAGs de monitorização e controlo dos cilindros pneumáticos e das janelas padrão, estão também associadas a variáveis membros das estruturas de dados criadas no PLC.

4.1.4. Teste ao programa

Nesta secção descreve-se os testes efetuados aos módulos de controlo e à interface desenvolvida para o primeiro caso de estudo. Para tal, os testes são realizados em ambiente de simulador do *Sysmac Studio*. Para simular o funcionamento do equipamento é necessário aceder às estruturas de dados dos módulos de controlo e alterar os valores de entrada, de modo a simular dados como a posição das paletes de transporte, a presença de garrafa na área de trabalho das estações e a condição de posicionamento das garrafas.

O caso de estudo apresentado é feito no modo de funcionamento "Automático" do equipamento pois, comparativamente à alternativa de funcionamento (Modo de funcionamento "Manual"), apresenta maior grau de complexidade a nível de controlo e trata-se do principal modo de funcionamento em que qualquer equipamento deste tipo a funcionar em ambiente empresarial. O modo de funcionamento "Manual" trata-se de um modo de funcionamento em que é o próprio

operador que aciona as funções desempenhadas pelo equipamento/estação, através de comandos específicos disponíveis nas diferentes janelas, tais como o monitor de controlo das estações, dos cilindros pneumáticos e do prato rotativo. O controlo manual possibilita assim ao utilizador realizar manobras de manutenção ou reparação do equipamento. Por estes factos, neste caso de estudo não se apresentará uma abordagem completa ao modo "Manual". Porém será apresentado um exemplo de um acionamento manual relativamente a um cilindro pneumático, num caso em que se coloca uma das estações em modo "Manual", no decorrer do ciclo automático da máquina.

Quando se inicializa a máquina, surge a janela principal da inteface, ver a Figura 53. Como já foi referido na secção 4.1.3, a janela principal dá acesso às janelas "Modo Funcionamento Geral", "Vista Geral", "Interface c/ Operador", "Estatísticas", "Configurações" e "Eventos".

Figura 53 - Ecrã Principal.

Após a visualização da janela principal, o utilizador deve começar por escolher um modo de funcionamento visto que não existe nenhum modo de funcionamento pré-definido (ver Figura 55). Para tal, deve entrar no ecrã "Modo de Funcionamento Geral" (ver Figura 54) e selecionar o modo de funcionamento com o qual pretende iniciar a máquina.

Figura 54 - Ecrã "Modo de Funcionamento Geral" – Caso Estudo 1.

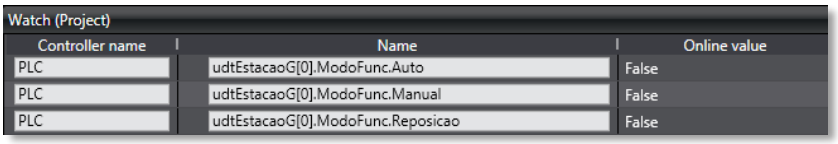


Figura 55 - Modos de funcionamento após iniciação da máquina.

Para iniciar o modo de funcionamento “Automático”, deve-se pressionar o botão “Iniciar” da janela “Modo de Funcionamento Geral”. Numa primeira fase a máquina entra em modo de “Reposição” (ver Figura 56), este modo de funcionamento coloca a máquina nas suas condições iniciais. Então este modo fica ativo até que se verifiquem as condições iniciais da máquina e, só depois dá início ao modo “Automático” (ver Figura 57).

Figura 56 – Modo de reposição (ecrã "Modo de Funcionamento Geral").

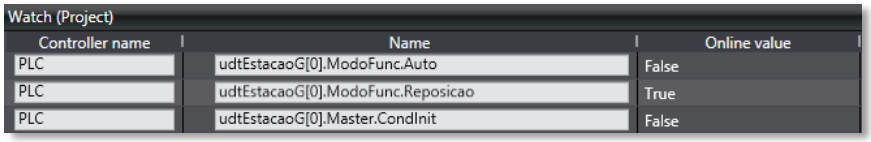


Figura 57 - Modos de funcionamento ("Reposição").

Assim que se verifiquem as condições iniciais (ver Figura 59), dá início ao modo de funcionamento "Automático" (ver Figura 58). De seguida, este modo de funcionamento permite iniciar o modo de "Produção" (ver Figura 60). O modo de "Produção", é um dos comandos específicos do equipamento, onde o operador apenas tem a função de monitorizar o processo, pois todas as estações realizam as suas tarefas automaticamente sem intervenção do operador.

Figura 58 – Modo automático (ecrã "Modo de Funcionamento Geral").

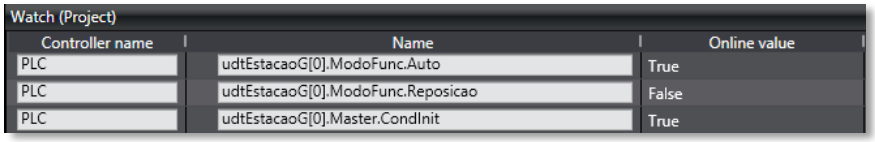


Figura 59 - Modos de funcionamento ("Automático").

Figura 60 – Modo de produção (ecrã "Modo de Funcionamento Geral).

Uma vez iniciado o modo de funcionamento "Automático" com o comando de "Produção", pode ser monitorizado o processo na janela de "Vista Geral" (ver Figura 61). Este ecrã, permite ao

operador aceder, através dos dois botões na lateral esquerda do ecrã (botões “Modo Funcionamento” e “Comandos Específicos”), às janelas de seleção do modo de funcionamento e os comandos específicos de cada estação. Permite ainda colocar a máquina em pausa e, posteriormente, recolocar novamente no modo automático através dos botões de acesso rápido (botão “Iniciar” e “Parar”).

Através da representação de um *Layout* da máquina, é possível monitorizar alguns estados da máquina como por exemplo: presença de garrafa, número da palete, estado dos *Stoppers* e o estado de funcionamento de cada estação. O estado de funcionamento das estações é apresentado através de um sinal luminoso que muda de cor. Quando a estação se encontra em produção apresenta a cor verde, quando está em pausa muda para a cor amarela e quando está à espera de peça muda para a cor cinzenta.

Nesta janela pode-se ainda monitorizar o estado do prato rotativo, da mesma forma da monitorização das restantes estações de trabalho para o estado de funcionamento. Esta janela permite também visualizar a ocupação dos quatro ninhos: caso o ninho não tenha garrafa apresenta a cor vermelha, caso esteja ocupado com garrafa terá a cor verde.

Esta janela permite ainda aceder à janela de interface com o operador através do botão na lateral esquerda do ecrã (botão “Janela de Interface”) onde o operador pode visualizar instruções a seguir e dados estatísticos do processo. Neste ecrã ainda existe a possibilidade de aceder às janelas de monitorização e controlo das estações. Para tal, basta pressionar em cima das estações dispostas no *layout* do equipamento.

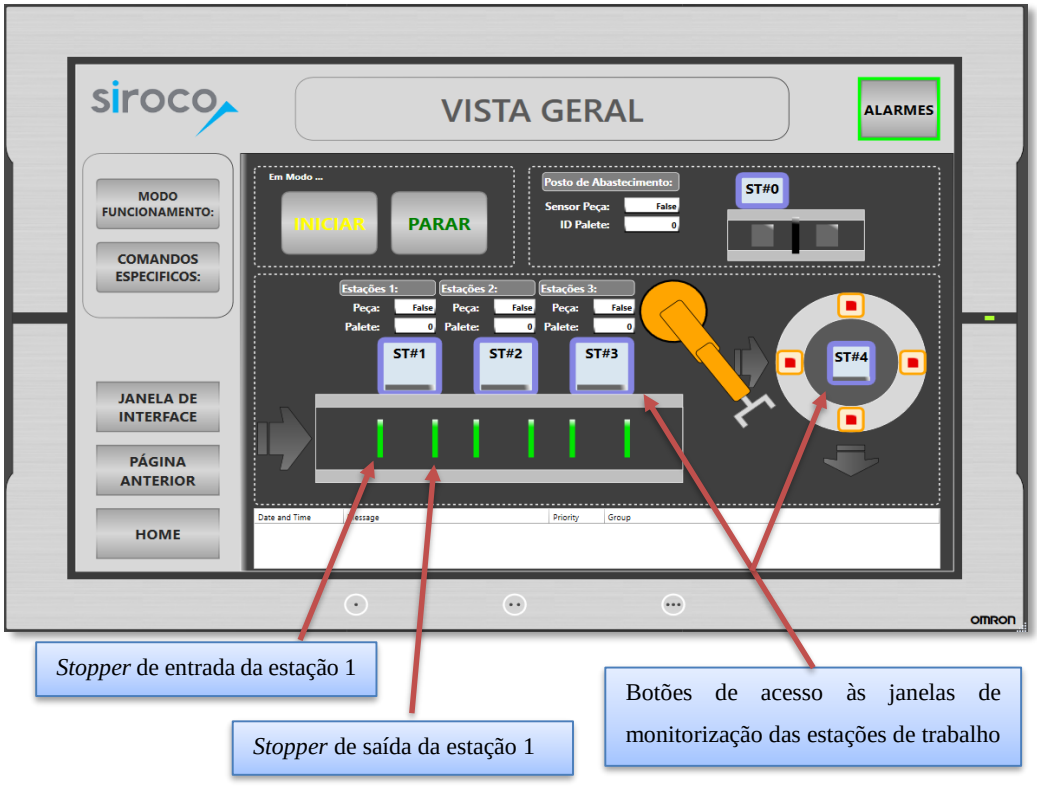


Figura 61 - Ecrã "Vista Geral" – Caso de Estudo 1.

O início de funcionamento da máquina dá-se com a entrada da primeira garrafa na linha (ver Figura 62 e Figura 63). Neste momento os *Stoppers* encontram-se todos desativados, exceto o

Stopper de saída da estação 1. Esta reação deve-se ao facto de a estação 1 ser a primeira a receber a garrafa. Como tal o *Stopper* já se encontra preparado para travar a palete na área de trabalho da estação 1. Todas as outras estações de trabalho encontram-se em "espera".

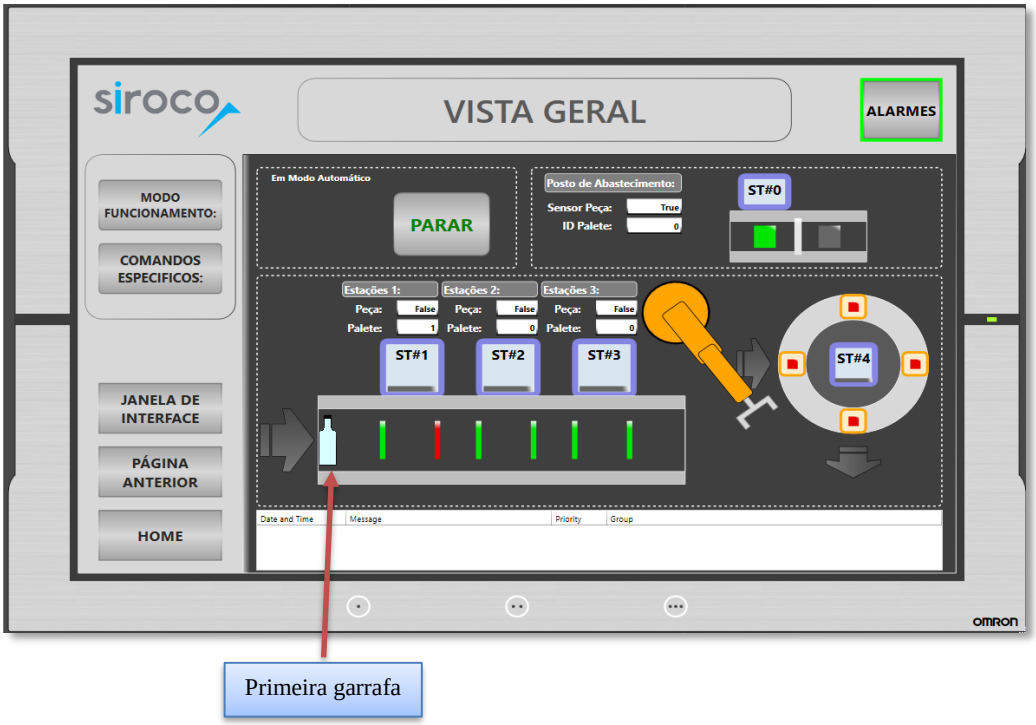


Figura 62 - Ecrã "Vista Geral" - entrada da primeira garrafa.

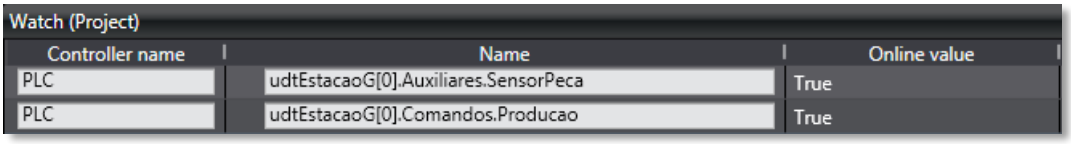


Figura 63 - Sensor de presença da garrafa no posto de abastecimento.

Após a entrada da primeira garrafa na linha, esta dirige-se para a estação 1. Assim que chega à área de operação (ver Figura 64) a garrafa passa pelo teste de verificação de posição (na simulação deste caso, para simplificar o processo, a variável que guarda o dado de verificação de posição da garrafa é sempre "forçada" a verdadeiro). Depois das verificações a estação 1 inicia a operação, ver Figura 65.

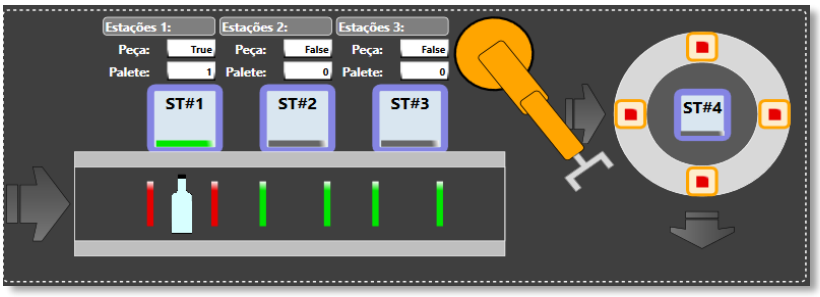


Figura 64 – "*Layout*" (ecrã "Vista Geral") - presença de garrafa da estação 1.

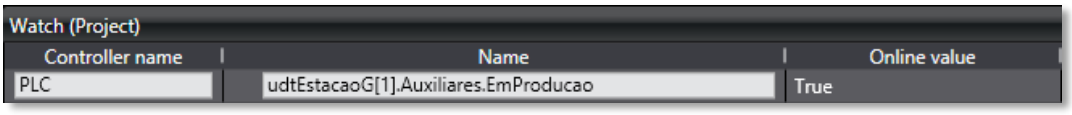


Figura 65 - Estação 1 em modo de "Operação".

Uma vez iniciada a operação da estação, pressionando no botão da estação 1, o processo pode ser acompanhado. O ecrã apresentado na Figura 66 é a janela de monitorização e controlo da estação 1. Nesta janela é possível acompanhar dados estatísticos (tempos de ciclo, número da palete, etc.) e o estado do processo da estação através do *layout* apresentado. Este ecrã apresenta o momento em que a estação 1 recebe a garrafa e dá início às operações (cilindro 0 e 1 em estado recuado).

Figura 66 - Ecrã de monitorização e controlo da estação 1.

Dado o início do processo da estação 1, o cilindro 1 avança para fixar a garrafa e de seguida avança o cilindro 2 para iniciar o sopro de limpeza na garrafa (ver Figura 67 e Figura 68).

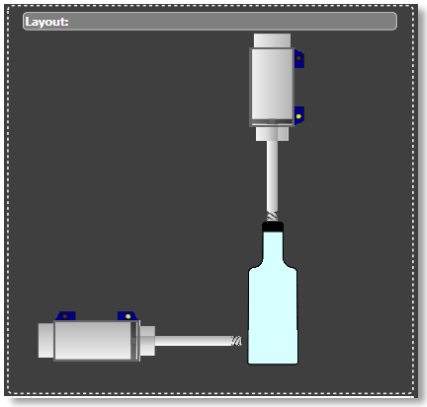


Figura 67 – "*Layout*" (ecrã de monitorização e controlo da estação 1) - (Avanço do cilindro 2).

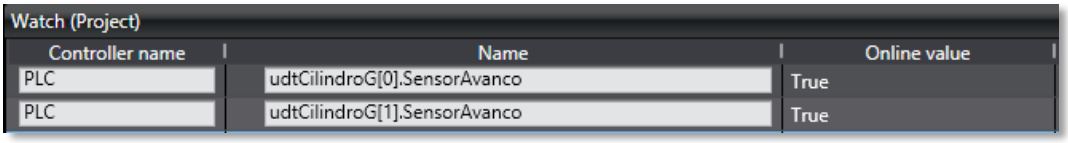


Figura 68 - Cilindro 1 e 2 Avançados.

No fim do tempo definido para a limpeza da garrafa, o cilindro 2 recua e de seguida o cilindro 1 também recua e termina o ciclo. Após o fim de ciclo de trabalho da estação 1, o *Stopper* de saída da estação recua permitindo a passagem da garrafa para a linha e permite também a entrada da próxima garrafa na estação 1 (ver Figura 69).

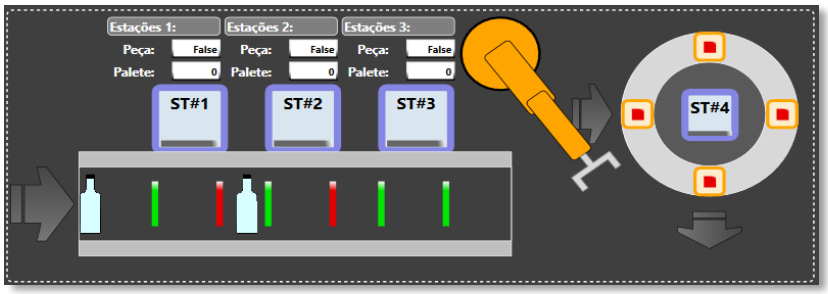


Figura 69 - "*Layout*" (ecrã "Vista Geral") - Entrada de uma nova garrafa.

Dada a entrada das garrafas (ver Figura 70) inicia-se o ciclo de trabalho nas estações 1 e 2. A estação 2 tem a função de encher a garrafa com um determinado líquido. Pode-se igualmente monitorizar o processo da estação 2 no ecrã de monitorização e controlo da estação 2, ver Figura 71.

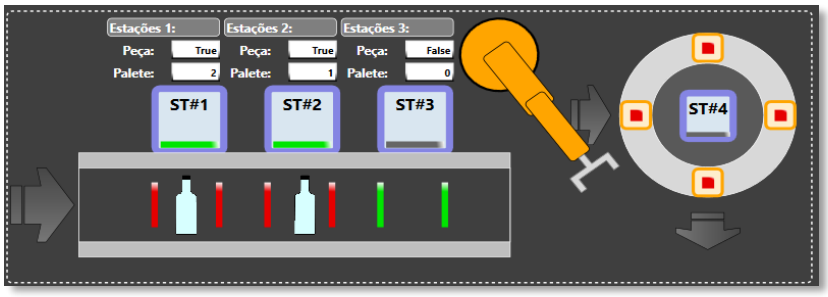


Figura 70 - "*Layout*" (ecrã "Vista Geral") - garrafa na estação 1 e 2.

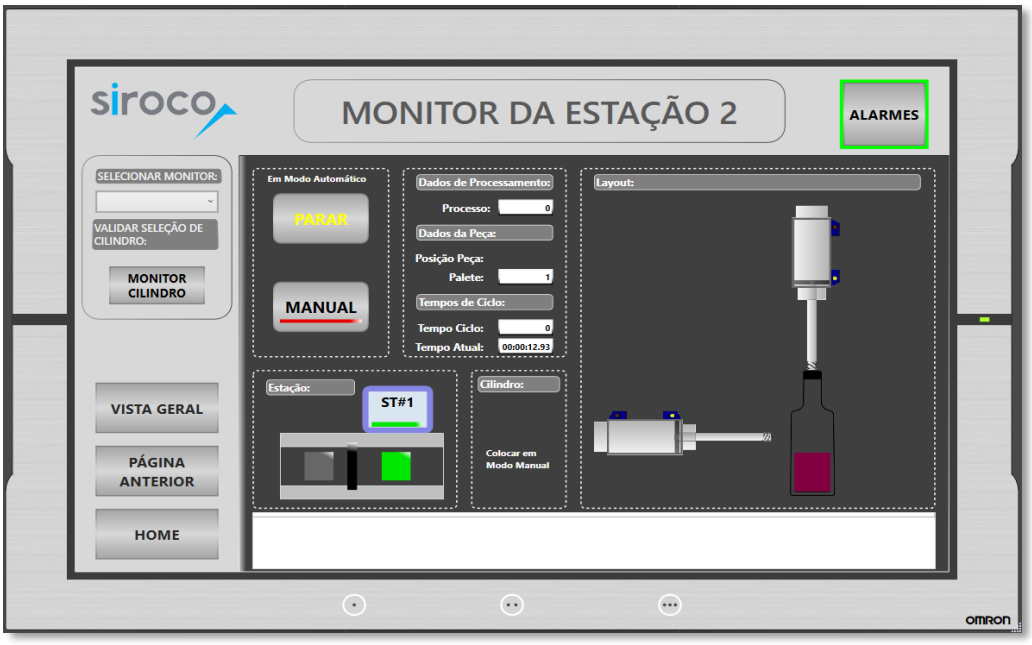


Figura 71 - Ecrã de monitorização e controlo da estação 1 - enchimento da garrafa.

Finalizados os processos nas estações 1 e 2, as garrafas entram novamente na linha e dirigem-se para a estação de trabalho seguinte. Após esse instante as três estações passam a operar em simultâneo e, num momento seguinte, quando a estação 3 finaliza o seu processo, a garrafa segue rumo à última estação da linha (prato rotativo), ver Figura 72.

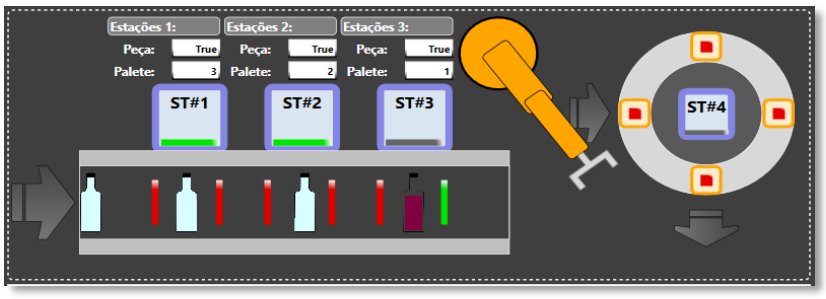


Figura 72 - "*Layout*" (ecrã "Vista Geral") - garrafa da palete 3 ruma ao prato rotativo.

No momento apresentado na Figura 73, a garrafa da palete 1 é colocada no primeiro ninho do prato rotativo. Pode-se confirmar que o ninho 1, encontra-se ocupado com uma garrafa e

posicionado no primeiro posto. O posto 1 é o posto de abastecimento do prato rotativo. Portanto assim que recebe garrafa, o prato roda no sentido horário, e dirige o ninho 1 até ao posto 2. No posto de trabalho 2 dá-se início à colocação do rótulo na garrafa.

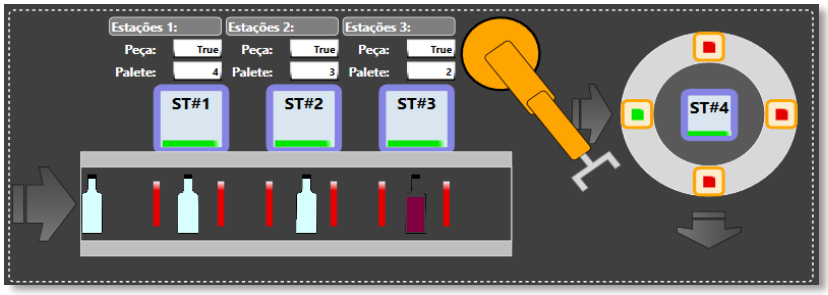


Figura 73 - "*Layout*" (ecrã "Vista Geral") - início de ciclo do prato rotativo.

Ao pressionar o botão da estação 4, é possível aceder ao ecrã de monitorização do prato rotativo e visualizar o estado atual do prato rotativo, ver Figura 74. Esta janela de monitorização do prato rotativo, permite ao utilizador visualizar alguns dados relativamente aos postos de trabalho e ninhos (Estado de funcionamento dos postos, localização e ocupação dos ninhos, etc.).

Figura 74 - Ecrã "Monitor do Prato Rotativo" – Caso Estudo 1.

O modo de funcionamento "Automático" da máquina permite ao operador operar manualmente uma estação. Isto é, o próprio operador ao selecionar este o modo de funcionamento "Manual" numa determinada estação, ver Figura 75, ele próprio terá que comandar e executar a

sequência de movimentos dos atuadores penumáticos da estação em causa, ver Figura 76. O processo também pode ser feito no ecrã de monitorização e controlo dos próprios atuadores pneumáticos.

Figura 75 - Ecrã "Controlo das estações" - Estação 1 em modo "Manual".

Figura 76 - Ecrã “Monitor da Estação 1”.

Existe ainda a possibilidade de colocar as estações em modo “*Bypass*”. Este modo de funcionamento tira uma determinada estação de serviço, ver Figura 77 (Estação 2). Em modo

"Automático, o operador tem a possibilidade de colocar o equipamento ou uma estação específica em "Pausa". Este estado de funcionamento também retira a estação de serviço (Estação 3).

Figura 77 - "*Modo de Funcionamento*" (ecrã "Controlo das Estações") – diferentes modos de funcionamento.

No caso de estudo apresentado fez-se a simulação do modo de funcionamento "Automático" (modo de produção) do equipamento. Ao simular o programa no modo de funcionamento automático, tem-se como objetivo analisar e demonstrar, com base nos dados analisados, o comportamento, autonomia e eficiência dos módulos de controlo desenvolvidos ("Estação de Trabalho", "Prato Rotativo" e "Cilindros Pneumáticos").

Na sequência de acontecimentos demonstrados nesta secção é possível verificar e validar o funcionamento e a intercomunicação entre os módulos de controlo. Neste caso de estudo, o programa é constituído por 8 *Function Block* de controlo das estações, 8 *Function Blocks* para o controlo dos cilindros pneumáticos, 1 *Function Block* de controlo do prato rotativo, 3 *Function Block* de controlo dos postos de trabalho, e 1 *Function Block* de controlo da linha de montagem. É importante também referir a capacidade de reutilização destes blocos funcionais e como são fundamentais para permitir o rápido desenvolvimento do programa.

4.2. Caso de Estudo 2 – *Secondary spool ASM equipment*

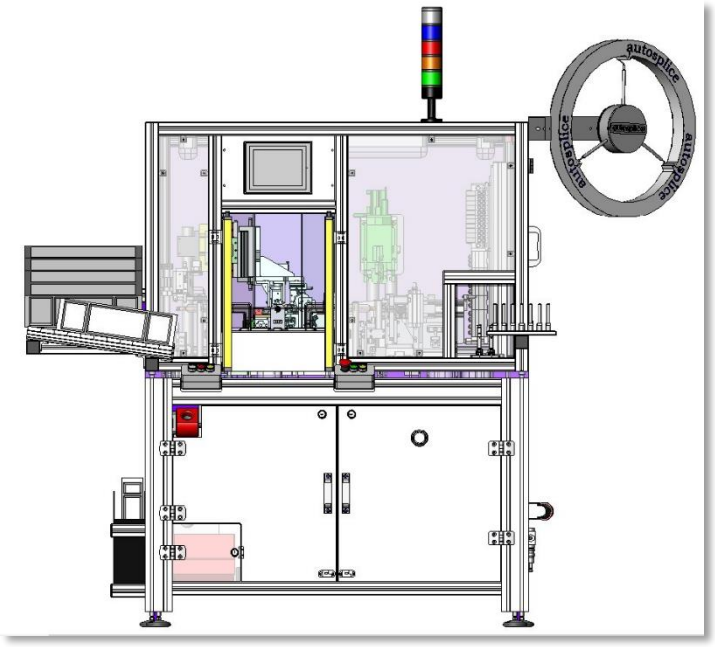


Figura 78 – Vista lateral do "*Secondary spool ASM equipment*".

4.2.1. Descrição geral

O equipamento real implementado pela empresa Siroco, foi a base de desenvolvimento do caso de estudo 2. O *Secondary Spool* ASM é o nome da peça que a máquina processa (ver Figura 79). A peça faz parte de um conjunto de ignição dos carros. Trata-se de um equipamento com o objetivo de inserir o terminal HV e díodo na peça *Secondary Spool* ASM.

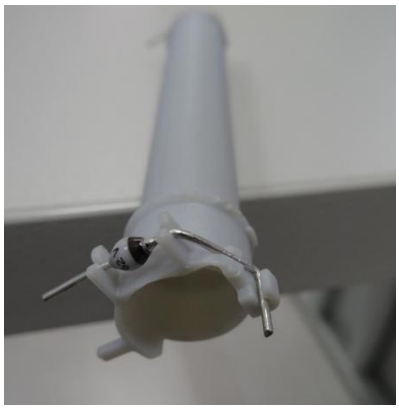


Figura 79 - *Secondary Spool* ASM.

A máquina é composta por trinta atuadores pneumáticos, um autómato programável, uma consola HMI e um prato rotativo. O prato rotativo é constituído por quatro postos e quatro ninhos. O primeiro posto é para carga/descarga do *secondary spool* (Posto 1 - Figura 80), o segundo é um posto de inserção e dobragem do terminal HV no *secondary spool* (Posto 2), o terceiro posto realiza o teste elétrico, corte, dobragem e inserção do díodo no *secondary spool* (Posto 3). Por último, o quarto posto faz a verificação dos terminais HV e LV dos díodos (Posto 4) de forma semiautomática

com intervenção de um operador apenas para alimentar a máquina com peças (*secondary spool*) e caso necessário alguma intervenção de acordo com algum alarme que ocorra.

O programa desenvolvido para o controlo deste equipamento será feito com recurso aos blocos funcionais desenvolvidos no projeto, nomeadamente a *Function Block* de controlo do prato rotativo, de controlo de estações e de cilindros pneumáticos.

Num primeiro momento, quando se inicializar o equipamento, os postos de trabalho estarão livres e à espera de peças. Sendo assim, a primeira peça colocada no posto de carga terá acesso direto ao segundo posto, caso esteja livre. Posteriormente a fluência de entrada de peças, estará sempre sujeita ao maior tempo de espera, isto é, o posto que tiver o maior tempo de ciclo, ditará a cadência do prato rotativo.

O processo de funcionamento dos postos de trabalho inicia-se com a chegada da peça às suas áreas de operação. Quando o posto de trabalho verifica que se encontra uma nova peça operável, confirma se a posição da peça no ninho é a correta, antes de dar início ao processo. A operação de verificação de posição é feita com recurso a sensores que verificam se existe algum mau posicionamento que possa pôr em causa o bom desempenho do processo. Sendo assim, os postos de trabalho apenas operam caso as peças se encontrem bem posicionadas. Caso contrário, a peça fica inoperável e não lhe é feita nenhuma operação até à saída do prato rotativo.

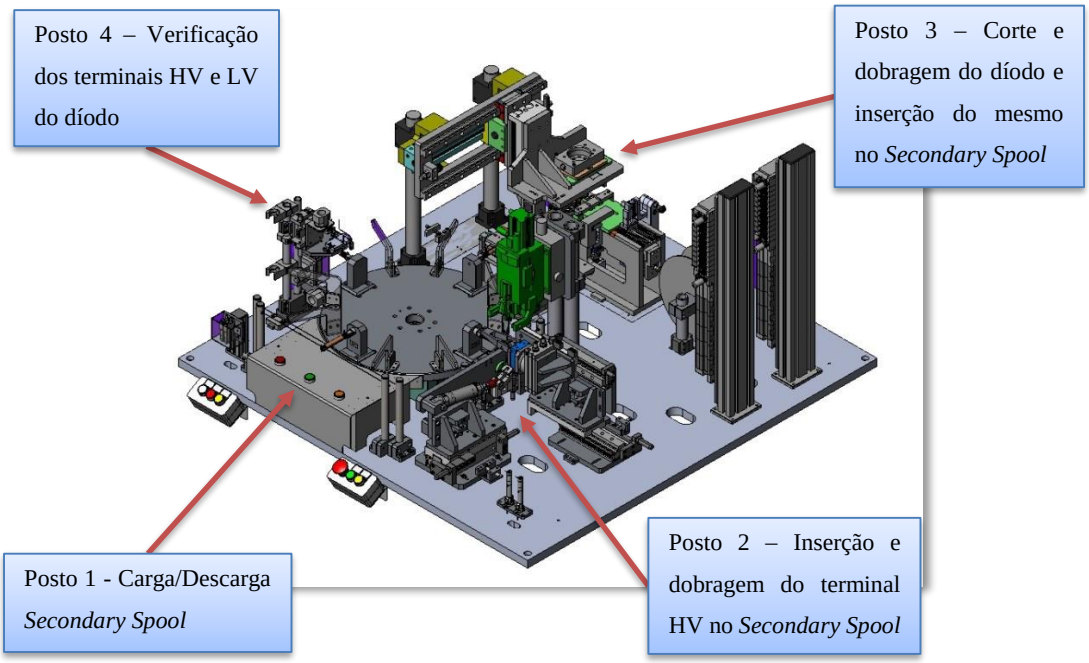


Figura 80 - *Layout* do Caso de Estudo 2.

A sequência de funcionamento do equipamento consiste:

- Posto 1 – Carga/Descarga do *secondary spool* – Operação manual;

- Posto 2 – Inserção e dobragem do terminal HV no *secondary spool* – Operação automática;
- Posto 3 – Teste elétrico (continuidade), separação (corte) do díodo da banda no alimentador, dobragem do díodo e inserção do díodo no *secondary spool* – Operação automática;
- Posto 4 – Verificação da presença do terminal HV e terminais LV do díodo inseridos no *secondary spool* – Operação automática;

Neste caso de estudo, não se dá foco à sequência de funcionamento dos postos de trabalho, uma vez que o funcionamento do controlo das tarefas das estações já foi referenciado no caso de estudo 1. Como tal, neste caso de estudo, apenas será apresentado o funcionamento do prato rotativo.

4.2.2. Criação dos POU

A organização escolhida para a criação dos POUs define a divisão dos programas em diferentes grupos de controlo. O primeiro grupo define o controlo geral da máquina, ou seja, as secções definidas neste grupo incluem as *Function Blocks* de controlo das estações (postos de trabalho). Seguem-se os grupos de controlo dos postos de trabalho, que serão quatro grupos, um para cada posto. As secções destes grupos de controlo dos postos incluem as *Function Blocks* de controlo dos cilindros pneumáticos, os programas de reposição, controlo e de verificação das permissões. Por fim, o grupo com as secções de controlo do prato rotativo: as secções deste último grupo incluem as *Function Blocks* de controlo do prato rotativo, o controlo dos postos de trabalho e as *Functions* de registo dos dados de leitura e contagem dos tempos de ciclo dos postos de trabalho. A Figura 81 permite visualizar a árvore de navegação dos POUs do programa desenvolvido para o controlo do equipamento.

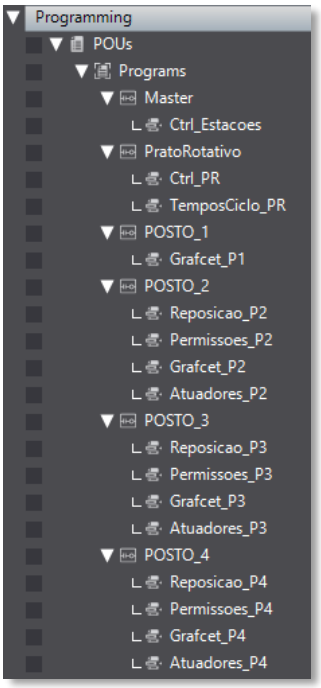


Figura 81 - Árvore de navegação dos POUs do caso de estudo 2.

4.2.3. Criação da interface

A interface Homem-Máquina do caso de estudo 2 foi desenvolvida de acordo com o modelo definido na secção 3.1. O desenvolvimento da interface deste caso de estudo seguiu a mesma linha de estrutura de ecrãs, da secção anterior, com algumas particularidades que podem ser verificadas no esquema da Figura 82 e Figura 83.

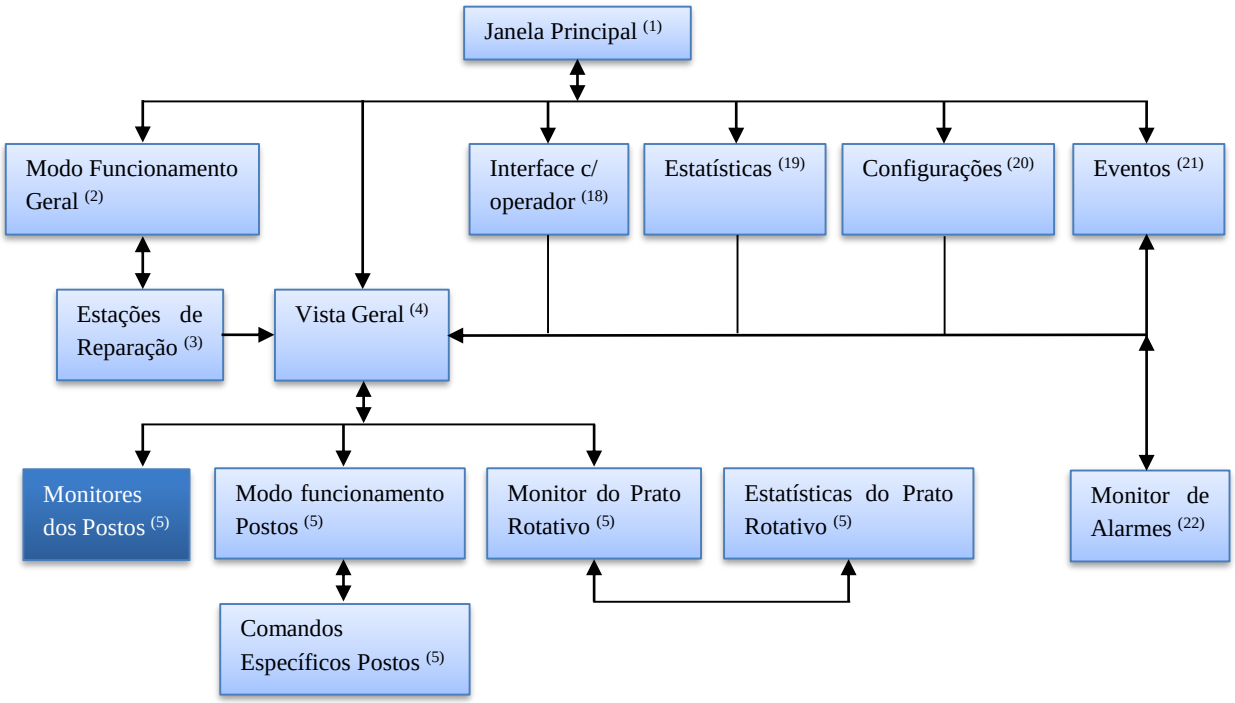


Figura 82 - Estrutura de ecrãs da consola do caso de estudo 2.

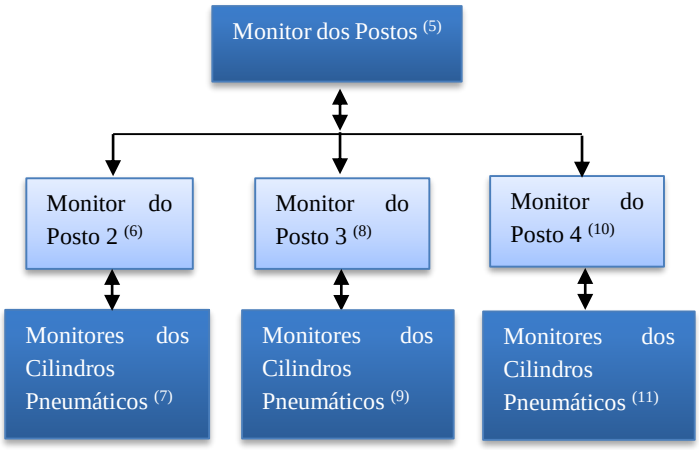


Figura 83 - Estrutura de ecrãs do grupo "Monitor dos Postos de Trabalho".

4.2.4. Teste ao programa

Nesta secção decreve-se o teste ao módulo de controlo do Prato Rotativo e à Interface desenvolvida para o caso de estudo 2. Para tal, os testes são realizados em ambiente de simulador do *Sysmac Studio*. Para simular o funcionamento do equipamento é necessário aceder às estruturas de dados dos módulos de controlo e alterar os valores de entrada, de modo a simular dados como a posição dos ninhos e a presença de peça no posto de carga.

O caso de estudo apresentado segue a mesma metodologia apresentada no caso de estudo 1 (secção 4.1.4) sendo também simulado no modo de funcionamento "Automático". O controlo manual possibilita assim ao utilizador realizar manobras de manutenção no prato rotativo, como a rotação do prato rotativo, ou controlo de postos de trabalho.

Uma vez inicializado o modo de funcionamento "Automático" com o comando de "Produção" pode ser monitorizado o processo na janela de "Vista Geral" (ver Figura 84). Este ecrã, à semelhança do caso de estudo 2, permite ao operador aceder, através dos dois botões na lateral esquerda do ecrã (botões "Monitor do Prato Rotativo" e "Modo Funcionamento Postos") à janela de monitorização do prato rotativo (ver Figura 86) e de seleção do modo de funcionamento e os comandos específicos de cada estação.

Através da representação de um *Layout* da máquina, é possível monitorizar alguns estados da máquina, como por exemplo a presença de peça no ninho, a posição do ninho relativamente ao posto de trabalho e o estado de funcionamento de cada posto.

Figura 84 - Ecrã "Vista Geral" – Caso de Estudo 2.

Com a introdução de uma peça no posto de carga, esta é transportada para o primeiro posto de trabalho (Posto 2). Quando o sensor de peça do posto 2 deteta a chegada de peça, o prato dá início ao ciclo de trabalho (ver Figura 85). A janela de monitorização do prato rotativo (ver Figura 86), permite demonstrar o funcionamento do posto 2 e a inserção de uma nova peça no posto de carga.

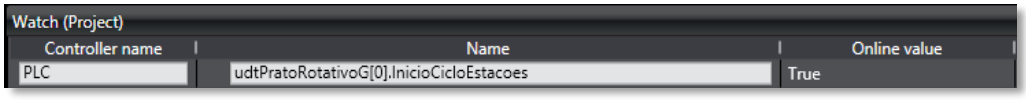


Figura 85 - Início de ciclo dos postos do prato rotativo.

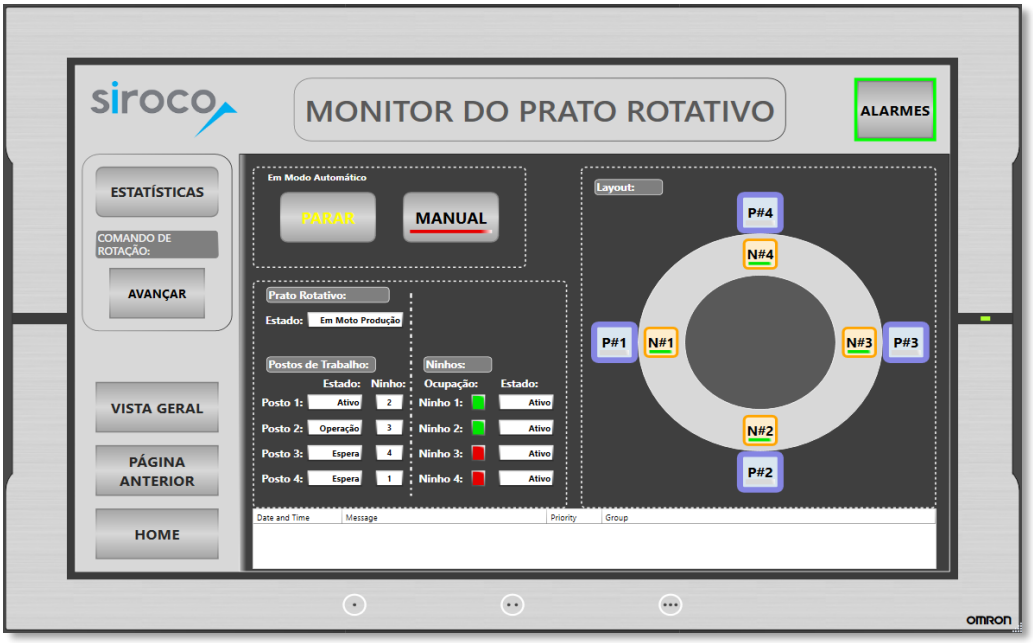


Figura 86 - Ecrã "Monitor Prato Rotativo" - Caso Estudo 2.

No fim de ciclo do posto 2, o prato rotativo inicia a rotação, ver Figura 87. Neste momento, como é possível visualizar na Figura 88, os postos 2 e 3 encontram-se ocupado e em operação, respetivamente. Ao aceder à janela dos detalhes estatísticos da produção do prato rotativo (ver Figura 89) é possível visualizar alguns dados estatísticos como os tempos de ciclo, dados de leitura (Teste de continuidade do posto 3) e peças produzidas.

Figura 87 - Início de rotação do prato rotativo.

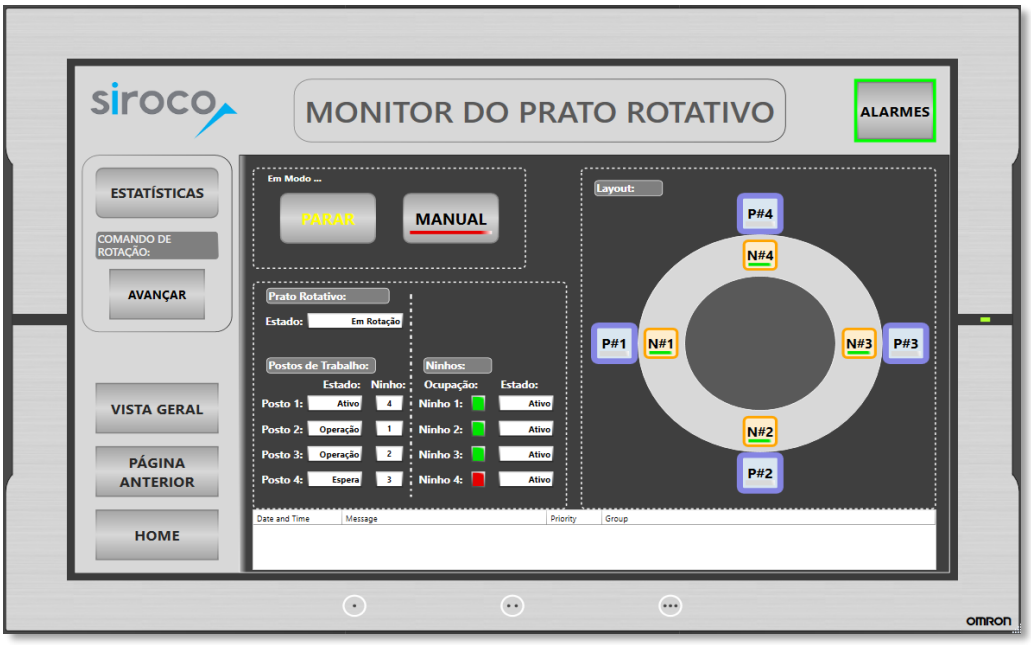


Figura 88 - Ecrã "Monitor Prato Rotativo", Postos 2 e 3 em operação.

Figura 89 - Ecrã "Detalhes da Produção" - Caso de Estudo 2.

Na sequência dos resultados apresentados nesta secção de testes ao programa, na simulação do programa desenvolvido para o caso de estudo 2, vem acrescentar aos factos já constatados no caso de estudo 1, que os blocos funcionais desenvolvidos apresentam caraterísticas de versatilidade, no que toca à sua aplicabilidade a novas situações.

4.3. Análise geral dos casos de estudo

Após terem sido desenvolvidos os diversos blocos de controlo, a implementação dos casos de estudo ensaiados foi rapidamente conseguida. É possível, deste modo, constatar que a abordagem modular é uma escolha acertada para programação de controladores. Isto porque, na implementação de um novo programa de controlo para um determinado equipamento, em vez da principal preocupação ser procurar um projeto semelhante que já tenha sido desenvolvido, passa a ser algo mais simples como começar por pensar que tipo de blocos e quantos serão necessários para controlar o processo, tornando assim todo o desenvolvimento de um programa mais rápido e intuitivo.

Sendo assim, a modularidade, flexibilidade e a boa gestão da complexidade de um algoritmo, é assegurada com a utilização de blocos de controlo. Dado que, o conceito de trabalhar com peças pequenas e pouco complexas, preparadas para serem acopladas numa maior é normalmente melhor do que usar diretamente uma peça grande com um elevado nível de complexidade. Com esta abordagem considera-se possível, em poucos passos, desenvolver um algoritmo de controlo de um equipamento, de forma rápida e organizada.

Por último, a prova de que é possível utilizar blocos funcionais na elaboração de programas de controlo de equipamento de montagem de componentes, é alcançada com a descrição das etapas realizadas nas seções 4.1.4 e 4.2.4. O facto de ter sido possível validar a utilização dos blocos funcionais na implementação dos dois casos de estudo, permite afirmar que a metodologia proposta é uma metodologia a considerar na implementação de sistemas de controlo/comando de máquinas e processos baseados em PLCs. E como tal, foi possível demonstrar que com a utilização dos blocos funcionais desenvolvidos neste projeto, a empresa terá vantagens económicas e competitivas, consequência da redução de tempo no desenvolvimento dos programas de controlo das máquinas.

5. CONCLUSÃO

Nos equipamentos de montagem de componentes o controlo/comando é, tipicamente, implementado em PLCs. Para empresas como a Siroco, dedicadas ao desenvolvimento deste tipo de equipamentos, o grau de repetibilidade de projetos é pequeno. Isso implica que, para cada novo projeto, seja necessário desenvolver um novo programa de PLC. Tipicamente, quando assim se trata, procura-se usar partes de código já desenvolvido. Contudo, este tipo de abordagem é de evitar devido às evidentes limitações quanto se trata da estruturação e modularização do *software*.

Surge a necessidade de optar por métodos diferentes. Como tal, a solução passa por uma abordagem que permita o desenvolvimento de módulos de controlo independentes e reutilizáveis que, interligados entre si, permitam ao programador implementar o controlo geral, com a vantagem que a configuração seja feita de forma mais simples e mais rápida.

Na procura de soluções mais eficientes, surge a proposta para o desenvolvimento deste projeto, resultado da parceria entre a empresa Siroco e a Universidade de Aveiro. Neste projeto a empresa propõe o desenvolvimento de blocos funcionais capazes de serem aplicados em soluções padrão de sistemas automatizados. Para tal, é necessário desenvolver uma metodologia capaz de aproveitar as potencialidades dos blocos funcionais para o controlo de módulos padrão, dos equipamentos tipo, desenvolvidos pela empresa Siroco.

Neste trabalho foi proposta uma metodologia para estruturar e implementar programas em PLC usando conceitos relacionados com técnicas de programação por objetos. Esta abordagem usa alguns conceitos implementados em diferentes linguagens de programação: objetos, atributos, classes e métodos. Uma "classe" é uma estrutura de *software* que encapsula um grupo de funções e atributos que estão relacionados entre si de algum modo, pode ser usada num programa, por meio das suas instancias, denominadas de "objetos". Os "objetos" possuem operações próprias, denominadas de "métodos". Os programas desenvolvidos através das definições de "classe", são intrinsecamente modulares, e permitem que grande parte do *software* desenvolvido seja reutilizável.

Este relatório descreve a implementação de uma metodologia baseada na filosofia da programação orientada por objetos, focando os aspetos estritamente relacionados com as caraterísticas dos *softwares* industriais de programação de autómatos e a necessidade de combinar diferentes métodos. Demonstram-se também as caraterísticas e as vantagens procedentes do uso da norma IEC 61131-3 na implementação da metodologia para a programação de autómatos. Nomeadamente na utilização de *Function Blocks*, *Functions* e linguagens equiparadas às linguagens de alto nível da programação de computadores, para facilitar a modulação e o encapsulamento de código, para que possa ser reutilizável.

Todavia, é importante lembrar que, na elaboração de um projeto, o desenvolvimento da interface Homem-Máquina é um processo demorado, tal como a elaboração do código do PLC. Para evitar o mesmo tipo de desvantagens que surgem no desenvolvimento dos algoritmos, adotaram-se métodos e ferramentas que permitem desenvolver e implementar objetos e janelas padrão reutilizáveis. Estes objetos foram implementados e aplicados com os mesmos princípios dos blocos funcionais. Desta forma, foi possível desenvolver objetos facilmente reutilizáveis e com a garantia

de eficiência na sua aplicabilidade. A opção de seguir a mesma abordagem usada anteriormente em objetos da interface veio trazer grandes vantagens no que toca à redução de tempo de desenvolvimento das interfaces Homem-Máquina.

Após a fase de desenvolvimento dos diversos blocos funcionais, é demonstrado como podem ser aplicados em casos reais e são identificadas algumas vantagens que estes blocos funcionais trarão no desenvolvimento de futuros projetos, nomeadamente a fácil adaptabilidade, a resposta eficiente, a rápida aplicabilidade e a possibilidade de os reutilizar. Para tal, são apresentados dois casos de estudo, o primeiro trata-se de uma "Linha de enchimento e rotulagem de garrafas" e o segundo caso "*Secondary spool* ASM *equipment*", trata-se de um equipamento desenvolvido pela empresa Siroco, onde o programa de controlo da máquina implementado segundo os seus métodos, foi refeito utilizando o método desenvolvido neste projeto.

Na sequência das etapas descritas nas secções 4.1.4 e 4.2.4 é possível verificar e validar o funcionamento e a intercomunicação entre os módulos de controlo. No caso de estudo 1, o programa é constituído por 8 *Function Block* de controlo das estações, 8 *Function Blocks* para o controlo dos cilindros pneumáticos, 1 *Function Block* de controlo do prato rotativo, 3 *Function Block* de controlo dos postos de trabalho, e 1 *Function Block* de controlo da linha de montagem. No caso de estudo 2, são novamente utilizadas as *Function Blocks* de controlo das estações de trabalho, dos cilindros pneumáticos e, do prato rotativo. É importante referir a capacidade de reutilização destes blocos funcionais e como são fundamentais para permitir o rápido e eficiente desenvolvimento do programa.

Como tal, pode-se constatar que a abordagem modular é uma escolha válida para programação de controladores, pois apresenta provas na implementação e simulação dos casos de estudo. Com estas constatações demonstradas pode-se afirmar que a abordagem implementada reduz o tempo no desenvolvimento de programas de controlo e reduz as probabilidades de erros, dado que os produtos desenvolvidos já foram testados noutras situações. Os blocos funcionais permitem acelerar o processo de desenvolvimento mais rápido e intuitivo de código.

Uma contribuição geral deste trabalho é uma abordagem inovadora para tratar o problema de projeto dos sistemas em causa. De acordo com os resultados obtidos conclui-se que seguir esta abordagem na programação de controladores trará vantagens à empresa, tais como a possibilidade de desenvolvimento de programas bem estruturados, a possibilidade de um controlo mais específico e eficaz de unidades mais pequenas e por isso mais facilmente tratáveis e, garantindo-se uma mais fácil reutilização do código.

Resumidamente, este método de desenvolver programas de autómatos, aumentará a eficiência no desenvolvimento de novos equipamentos, possibilitando a redução de custos de desenvolvimento e manutenção.

Todavia, o desenvolvimento dos vários blocos funcionais com vista a obter soluções padrão mostrou ser um processo complexo e com elevado grau de abstração. Como grande parte do código se encontra subdividido em pequenas partes, torna-se difícil seguir o raciocínio lógico do programa, tanto na sua construção como na sua compreensão. Devido a este facto, torna-se imperativo a existência de uma boa documentação.

Outro facto importante é que a metodologia proposta representa uma forma diferente da tradicional de programação de PLCs e, por isso, deverá ainda ser testada em situações mais

complexas para avaliar de uma forma mais sustentada os ganhos de produtividade em relação ao nível de complexidade na elaboração dos programas.

Face aos resultados obtidos pode-se concluir que foram atingidos todos os objetivos propostos para a realização deste projeto. Nomeadamente:

- Desenvolvimento de um algoritmo de controlo de estações;
- Desenvolvimento de um algoritmo de controlo dos cilindros pneumáticos;
- Desenvolvimento de um algoritmo de controlo do prato rotativo;
- Integração dos alarmes/erros de utilizador utilizando o PLC NJ da *Omron*;
- Desenvolvimento de Interfaces HMI padrão para o controlo e monitorização das estações, cilindros, prato rotativo e de eventos.

5.1. Proposta de Trabalhos Futuros

Uma vez demonstrada a potencialidade do uso de blocos funcionais para elaboração de programas de PLCs, será uma ideia interessante para trabalhos ou projetos futuros a empresa apostar no desenvolvimento de módulos de controlo de outros equipamentos e sistemas de produção como, por exemplo, módulos de controlo de comunicação, controlo de manipuladores industriais, sistemas de visão computacional, entre outros equipamentos desenvolvidos pela empresa.

No que diz respeito à programação da interface HMI, tendo em conta as potencialidades da metodologia utilizada para desenvolver objetos e janelas padrão, há ainda a possibilidade de vir a desenvolver mais objetos deste tipo, nomeadamente para a monitorização dos alarmes ou outras estruturas que não tenham sido tratadas neste projeto. Tendo em conta que é possível incluir ficheiros multimédia para o ambiente da interface, seria importante tirar partido desta potencialidade ao nível, por exemplo, da existência de vídeos e documentação exemplificativos da resolução de erros/alarmes que permitam ao operador uma tomada de decisão mais rápida e eficiente.

Para tornar o equipamento final mais eficiente, a empresa deverá continuar a apostar no desenvolvimento de soluções de gestão e processamento automático de falhas/anomalias mais completos e intuitivos para, dessa forma minimizar o número de intervenções necessárias, bem como uma análise/reação mais fácil e rápida por parte dos operadores.

Para complementar todo o trabalho desenvolvido é ainda importante salientar que, em trabalhos futuros, a criação e desenvolvimento de esquemas elétricos modulares e escaláveis assumirá um papel fulcral como parte integrante neste tipo de projetos, visto que possibilita uma redução de tempo na elaboração dos esquemas elétricos do equipamento.

6. REFERÊNCIAS

[1] M. Bonfe and C. Fantuzzi, "Object-oriented approach to PLC software design for a manufacture\nmachinery using IEC 61131-3 norm languages," *2001 IEEE/ASME Int. Conf. Adv. Intell. Mechatronics. Proc. (Cat. No.01TH8556)*, vol. 2, no. July, pp. 787–792, 2001.

[2] T. Simon and S. Rosch, "Comparing the object-oriented extension with the classical IEC 61131-3 regarding reusability and understandability - A case study," *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, vol. 2015-Octob, 2015.

[3] M. Wilson, "Automation System Components," *Implement. Robot Syst.*, pp. 39–73, 2015.

[4] M. P. Groover, *Production Systems , and Manufacturing edition*. 2015.

[5] R. Steinberg, "Assembly Line Balancing," *Oper. Manag.*, vol. 3, no. 3, pp. 1–23, 2007.

[6] N. Boysen, M. Fliedner, and A. Scholl, "Assembly line balancing: Which model to use when?," *Int. J. Prod. Econ.*, vol. 111, no. 2, pp. 509–528, 2007.

[7] C. Becker and A. Scholl, "A survey on problems and methods in generalized assembly line balancing," *Eur. J. Oper. Res.*, vol. 168, no. 3, pp. 694–715, Feb. 2006.

[8] J. N. Pires, *Automação Industrial*, 3ª edição. 2007.

[9] P. Of and L. Control, *INDUSTRIAL AUTOMATION PRACTICES - Specification and programming of logic control applications in the " Its PLC " training environment*, 1ª Edição. Real Games, 2012.

[10] C. C. De Moraes and P. D. L. Castrucci, *Engenharia de Automação Industrial*, 2ª Edição. LTC - GRUPO GEN; , 2007.

[11] A. Gomes, "WEBGRAF - Aplicação Web para Execução de GRAFCETs e Redes de Petri em Controladores Lógicos Programáveis." 2003.

[12] J. Machado, "Concepção e Realização do Comando Operacional de Sistemas Industriais de Eventos Discretos," 2001.

[13] W. Bolton, *Programmable Logic Controllers*, 5 edition., vol. 1. Newnes, 2009.

[14] D. Weidm, "Automação Industrial – U-Remote," pp. 1–118, 2014.

[15] L. A. Bryan and E. A. Bryan, *Programmable Controllers: Theory and Implementation*. Industrial Text Co; 2nd edition (January 1997), 1997.

[16] F. Bonfatti, P. Monari, and U. Sampieri, *IEC 1131-3 programming methodology*, Chris Hart. CJ International, 1999.

[17] A. Malinowski and H. Yu, "Comparison of Embedded System Design for Industrial Applications," *IEEE Trans. Ind. Informatics*, vol. 7, no. 2, 2011.

[18] R. W. Sonnenfeldt, "Sumnary."

[19] G. C. G. Chao, "Human-Machine Interface: Design Principles of Visual Information in Human-Machine Interface Design," *2009 Int. Conf. Intell. Human-Machine Syst. Cybern.*, vol. 2, pp. 262–265, 2009.

[20] S. Mackay, E. Wright, D. Reynders, and J. Park, "Practical Industrial Data Networks: Design, Installation and Troubleshooting," p. 448, 2004.

[21] A. Storr and D. Jarvis, Eds., *Software Engineering for Manufacturing Systems*. Springer Science+Business Media Dordrecht, 1996.

[22] R. Zurawski and M. Zhou, "Petri Nets and Industrial Applications: A Tutorial," *IEEE Trans. Ind. Electron.*, vol. 41, no. 6, pp. 567–583, 1994.

[23] R. David and A. Hassane, "Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems," *Prentice-Hall*, vol. 30, no. 2, 1992.

[24] A. M. Reyna, A. G. Ortega, N. S. Romero, D. A. Diaz, S. E. F. Murillo, G. A. Felix Zarate, and S. L. N. Granados, "Object-oriented programming as an alternative to industrial control," *CCE 2012 - 2012 9th Int. Conf. Electr. Eng. Comput. Sci. Autom. Control*, 2012.

[25] A. Apostolov and S. Monnier, "Object-oriented design of human-machine interface for substation integration systems," *Electr. Comput. Eng.*, 1997.

[26] MK Technology Group, “Stopper and separator at feed via belt conveyor.” [Online]. Available: http://www.mk-group.com/en/references/transfer-system-references.html. [Accessed: 22-Jun-2016].
[27] F. Ebel and M. Pany, “Bestimmungsgemäße Verwendung / Intended use,” Denkendorf, 2006.

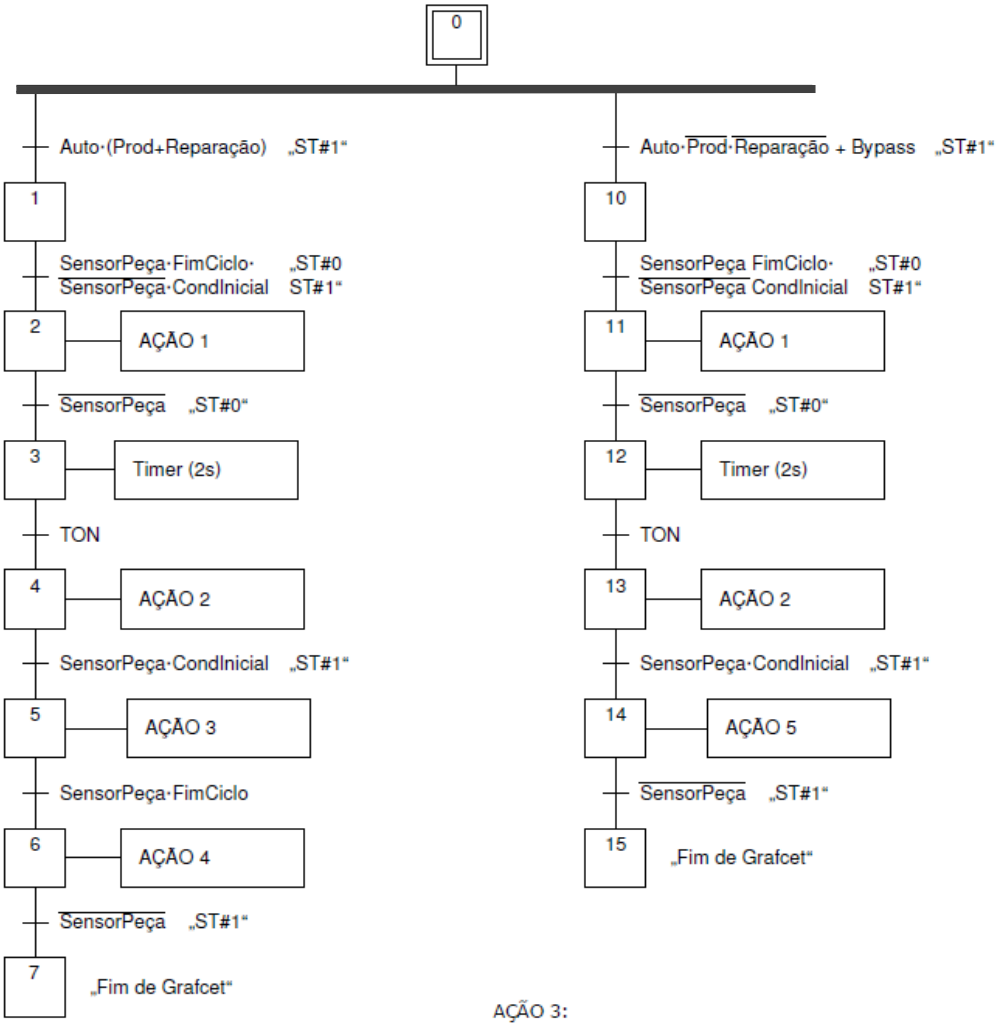
ANEXOS

Anexo 1 – GRAFCET do controlo do funcionamento das estações numa linha de montagem.

O GRAFCET apresentado, escreve a sequência de procedimentos no funcionamento das estações no modo automático, com os comandos de produção ou reparação e, o funcionamento das estações no modo bypass.

No primeiro caso, se a estação estiver em modo automático e com um dos comandos anteriormente mencionados ativos, o processo é responsável pelo controlo do funcionamento das operações da estação quando esta recebe uma peça pronta a ser processada.

No segundo caso, se a estação estiver em modo bypass, a estação recebe e retém a peça, sem efetuar algum processo, até que haja condições para a peça regressar à linha de transporte.



AÇÃO 1:
- udtEstacao1.Auxiliares.StopperOff [SET];
- udtEstacao2.Auxiliares.StopperOn [SET];
- udtEstacao1.Auxiliares.StopperOn [RESET];
- udtEstacao2.Auxiliares.StopperOff [RESET];

AÇÃO 2:
- udtEstacao1.Auxiliares.StopperOn [SET];
- udtEstacao1.Auxiliares.StopperOff [RESET];
- udtEstacao1.Auxiliares.FimCiclo [RESET];

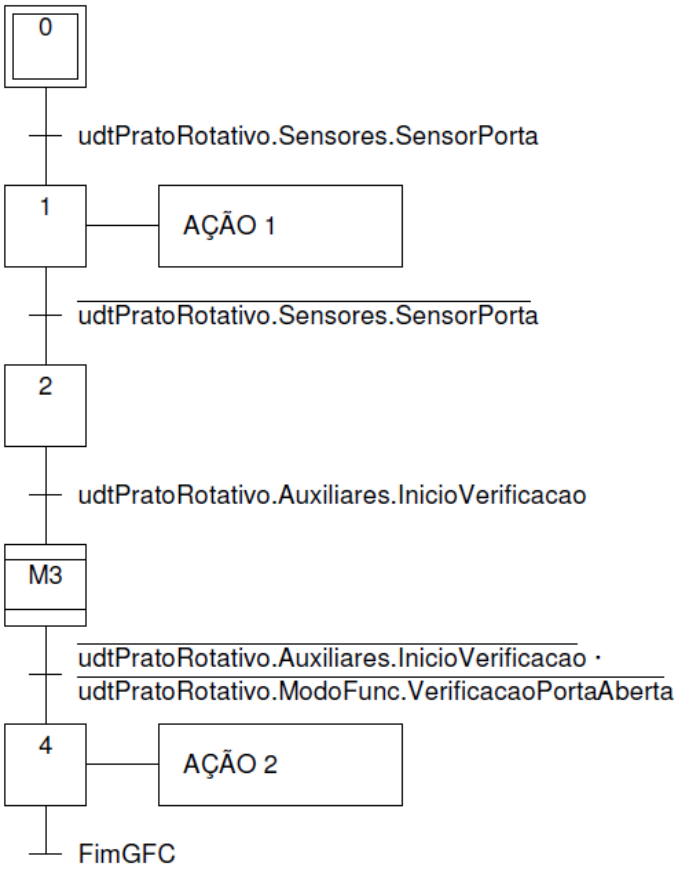
AÇÃO 3:
- udtEstacao2.Auxiliares.InicioCiclo [SET];
- udtEstacao1.Auxiliares.PecaProcessada [RESET];

AÇÃO 4:
- udtEstacao2.Auxiliares.PecaProcessada [SET];
- udtEstacao2.Auxiliares.InicioCiclo [RESET];

AÇÃO 5:
- udtEstacao2.Auxiliares.FimCiclo [SET];
- udtEstacao2.Auxiliares.PecaProcessada [SET];
- udtEstacao1.Auxiliares.PecaProcessada [RESET];

Anexo 2 - GRAFCET de rearme da porta do prato rotativo.

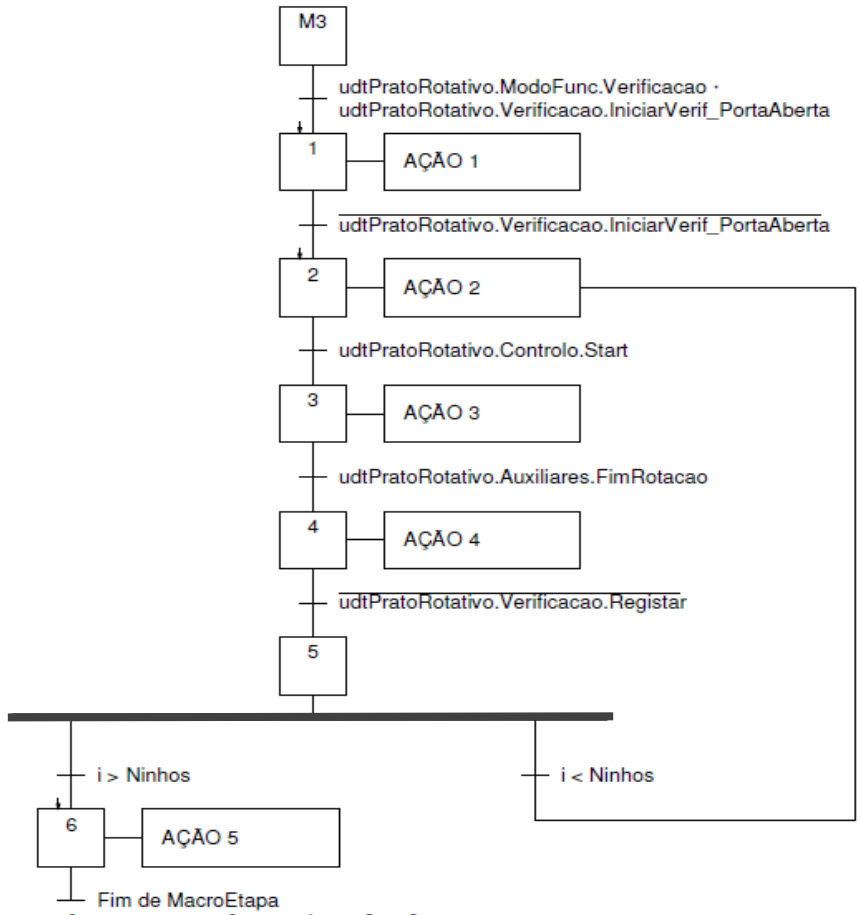
O GRAFCET apresentado descreve a sequência de procedimentos do rearme da porta do prato rotativo. Se a porta de proteção da estação do prato rotativo for aberta pelo operador durante o modo de produção, o processo principal do prato rotativo fica suspenso e, após o fecho da porta é necessário proceder ao rearme do funcionamento. Este processo é responsável por verificar se as condições dos ninhos (ocupação dos ninhos) foram alteradas durante o tempo em que a porta esteve aberta.



AÇÕES:
1 -udtPratoRotativo.ModoFunc.VerificacaoPortaAberta [SET];
2 -udtPratoRotativo.ModoFunc.VerificacaoPortaAberta [RESET].

Anexo 3 - GRAFCET de verificação dos ninhos do prato rotativo.

O GRAFCET apresentado, descreve a sequência de procedimentos da verificação dos ninhos do prato rotativo. Neste processo faz se a verificação de todos os ninhos após o rearme da porta do prato rotativo. Caso alguma das condições dos ninhos do prato tenha sido alterada durante o período de tempo em que a porta esteve aberta, por exemplo, peça retirada de um dos ninhos que estava ocupado, o processo de verificação despoleta um erro, que notifica a ocorrência na janela de alarmes.



AÇÃO 1:
- udtPratoRotativo.Verificacao.InicioVerif_PortaAberta [SET];
- udtPratoRotativo.Verificacao.IniciarVerif_PortaAberta [RESET];

AÇÃO 2:
- udtPratoRotativo.Verificacao.Registar [SET];
- i++;

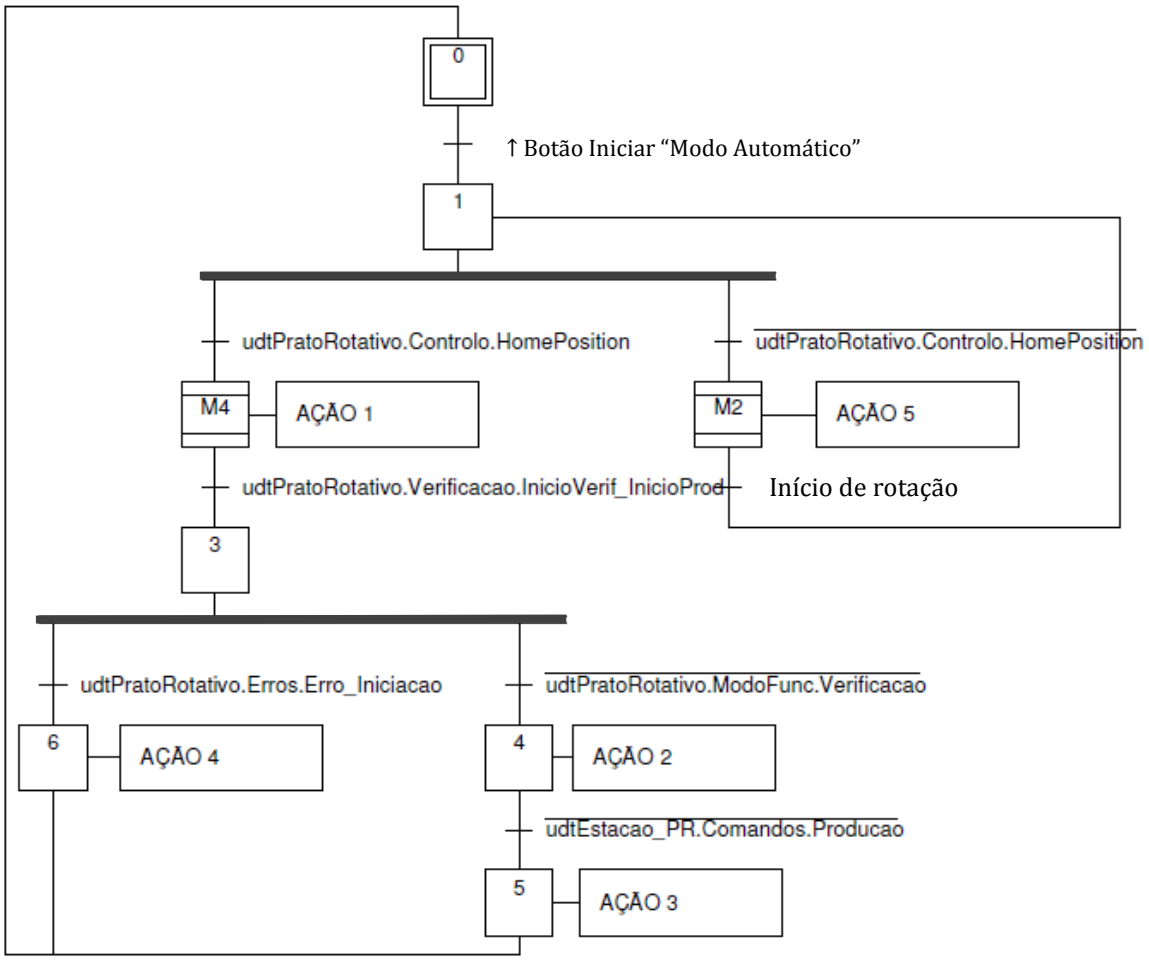
AÇÃO 3:
- udtPratoRotativo.RodaVerificacao [SET];

AÇÃO 4:
- udtPratoRotativo.RodaVerificacao [RESET];

AÇÃO 5:
- udtPratoRotativo.Verificacao.FimVerif [SET];
- i=1;

Anexo 4 – GRAFCET de iniciação modo de produção do prato rotativo.

O GRAFCET apresentado, descreve a sequência de procedimentos da iniciação do modo de produção do prato rotativo. Neste processo faz se a verificação de todos os ninhos e, caso algum dos ninhos esteja ocupado por uma peça, o processo despoleta um alarme. Após a verificação, o processo roda o prato até à posição inicial. Nesta posição o primeiro ninho (ninho com a identificação "1") fica alinhado com o primeiro posto de trabalho do prato rotativo.



AÇÃO 1:
- udtPratoRotativo.ModoFunc.Verificacao [SET];
- udtPratoRotativo.Verificacao.IniciarVerif_InicioProd [SET];

AÇÃO 2:
- udtPratoRotativo.Auxiliares.ModoProducao [SET];

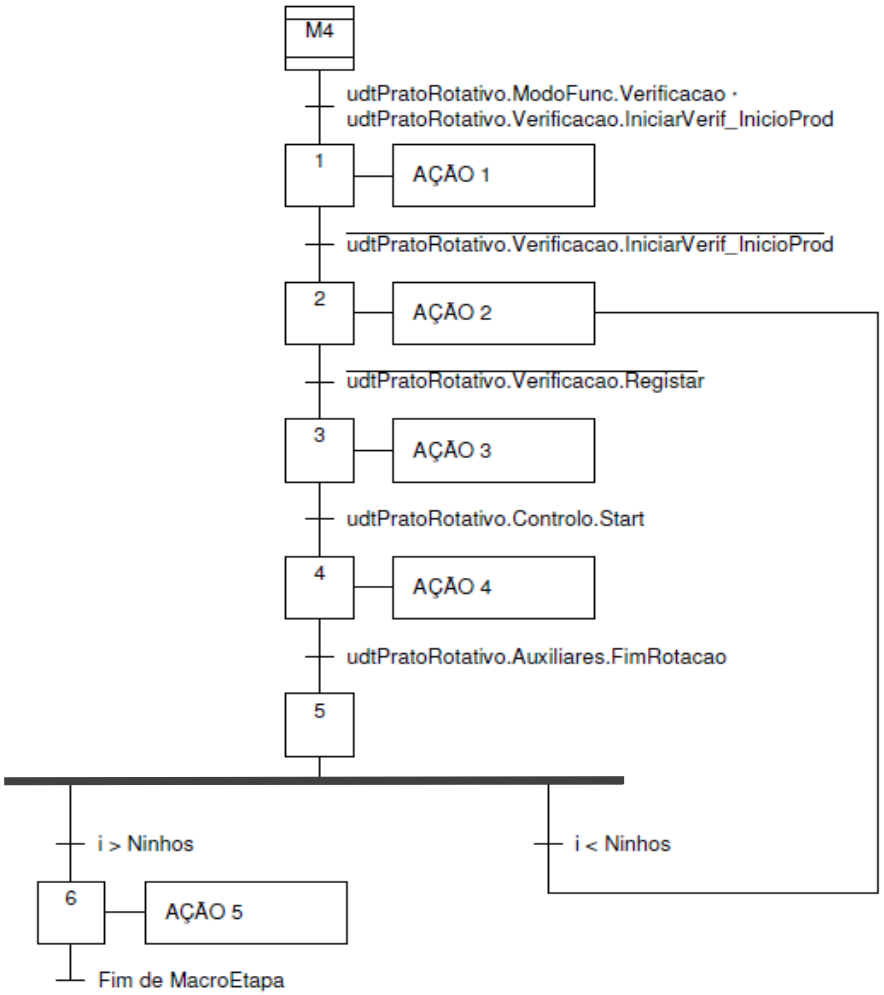
AÇÃO 3:
- udtPratoRotativo.Auxiliares.ModoProducao [RESET];

AÇÃO 4:
- udtEstacao_PR.Comandos.Producao [RESET];

AÇÃO 5:
- udtPratoRotativo.RodaReposicao;

Anexo 5 – GRAFCET de verificação dos ninhos do prato rotativo.

O GRAFCET apresentado, descreve a sequência de procedimentos da verificação dos ninhos do prato rotativo. Neste processo faz se a verificação de todos os ninhos necessária para a iniciação do modo de produção do prato rotativo. Caso alguma das condições dos ninhos do prato tenha sido alterada durante o período de tempo em que a porta esteve aberta, por exemplo, peça retirada de um dos ninhos que estava ocupado, o processo de verificação despoleta um erro, que notifica a ocorrência na janela de alarmes.



AÇÃO 1:
- udtPratoRotativo.Verificacao.InicioVerif_InicioProd [SET];
- udtPratoRotativo.Verificacao.IniciarVerif_InicioProd [RESET];

AÇÃO 2:
- udtPratoRotativo.Verificacao.Registar [SET];
- i++;

AÇÃO 3:
- udtPratoRotativo.RodaVerificacao [SET];

AÇÃO 4:
- udtPratoRotativo.RodaVerificacao [RESET];

AÇÃO 5:
- udtPratoRotativo.Verificacao.FimVerif [SET];
- i=0;